НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

«На правах рукопису»
УДК 004.043

«До захисту допущено»

Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
“___”_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: ____Інтелектуальна система перетворення чорно-білого зображення

Виконав: студент другого курсу, групи ІТ-84мп
(шифр групи)

Стельмах Олег Олександрович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., доцент Корнага Я.І. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____НК_____ ____к.т.н., доцент, Пасько В.П._____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент __д.т.н., проф. , професор каф. ММСА, Мухін В.Є.____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Стельмаху Олегу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Інтелектуальна система перетворення чорно-білого зображення»,
_____
науковий керівник дисертації доцент, к.т.н., доцент Корнага Я. І.,_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 2019 р. №_____

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження – комп'ютерне бачення для перетворення монохромного зображення в кольорове.

4. Предмет дослідження – реконструювання чорно-білого зображення.

5. Перелік завдань, які потрібно розробити – аналіз проблеми та існуючих рішень; аналіз і реалізація алгоритму; розробка програмного забезпечення; дослідження ефективності розробленого моделі програмного забезпечення.

6. Орієнтовний перелік ілюстративного матеріалу – три плакати

7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Аналіз предметної області	20.09.2019 р.	
2	Постановка задачі	30.09.2019 р.	
3	Аналіз інформаційного забезпечення	15.10.2019 р.	
5	Аналіз алгоритмічного забезпечення	28.10.2019 р.	
6	Розробка алгоритмічного забезпечення	10.11.2019 р.	
7	Розробка програмного забезпечення	25.11.2019 р.	
8	Маркетинговий аналіз стартап-проекту	10.12.2019 р.	
9	Висновки	15.12.2019 р.	

Студент _____ О. О. Стельмах
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Я. І. Корнага
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

У роботі розглянуто проблему в області автоматизованого перетворення чорно-былого зображення, показано основні особливості існуючих рішень та додатків, їх переваги та недоліки.

Для обробки зображення була розроблена нейронна мережа кольоризації, також був розроблений модуль покращення якості зображення. Дана система надає користувачу гнучкі можливості вибору алгоритмів з якими він бажає працювати.

Визначено завдання для системи перетворення чорно-білого зображення за допомогою нейронної мережі та відібрано нейронну мережу та спосіб навчання, які найбільш підходять для даної задачі. Описано структуру системи та проведено експерименти ефективності її роботи.

Ключові слова: нейронна мережа, машинне навчання, інтелектуальна система, кольоризація, чорно-біле зображення.

Розмір пояснювальної записки – 80 аркушів, містить 22 ілюстрації, 24 таблиць, 5 додатків.

ABSTRACT

The paper deals with the problem of automated conversion of black and former image, shows the main features of existing solutions and applications, their advantages and disadvantages.

A neural network of colorization was developed for image processing and a module for improving image quality was also developed. This system gives the user the flexibility to choose which algorithms he wants to work with.

The tasks for the system of transformation of black and white image by means of neural network are determined and the neural network and the method of training that are most suitable for this task are selected. The structure of the system is described and experiments of its work efficiency are conducted.

Keywords: neural network, machine learning, intelligent system, colorization, black and white image.

The size of the explanatory note - 80 sheets, contains 22 illustrations, 24 tables, 5 appendices.

Пояснювальна записка

до магістерської дисертації

на тему: *Інтелектуальна система перетворення чорно-білого зображення*

Київ – 2019 ро

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CNN – згорткова нейронна мережа

ГІС – геоінформаційні системи

ПЗ – програмне забезпечення

IoU – індекс перетину

ВСТУП

Одними з найважливіших областей досліджень і розробок сучасної кібернетики є області машинного навчання, розпізнавання образів і комп'ютерного зору. Прискорюються темпи розвитку технологій інформаційного суспільства, розвиток робототехніки, розвиток концепцій «розумний будинок» і «розумне місто», розвиток інтернету речей і систем штучного інтелекту визначають цій області особливе місце в сучасному науковому знанні. У багатьох прикладних задачах в практиці сучасного програмування використовуються методи збору даних, кластеризації та класифікації, методи статистичного висновку. У повсякденному житті, як і в корпоративну, і в промислову середовища починають впроваджуватися технології, поступово стираючи грань між реальним і віртуальним простором.

Завдання комп'ютерної графіки полягає в обробці інформації, пов'язаної з зображеннями. Вона розділяється на три основних напрямки: візуалізація, обробка зображень і розпізнавання зображень.

Обробка зображень – будь-яка форма обробки інформації, для якої вхідні дані представлені зображенням і результат – теж зображенням.

Завданням обробки може бути як поліпшення зображення до якогось певного критерію, так і перетворення, що кардинально міняє зображення. Різні методи застосовуються для оцифрованих зображень (фотографій, відео) і зображень, отриманих за допомогою комп'ютерної візуалізації. У разі оцифровки можна помітити шум на зображенні.

Важливою і актуальною проблемою досі є відновлення та перетворення фотографій після їх оцифровки, також ще є необхідність у видаленні різного роду шумів, подряпин і інших дефектів. завдання відновлення зображення полягає в тому, щоб по спостережуваному зображенню знайти більш повні характеристики вихідного об'єкта.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Об’єкт та предмет дослідження

Багато хто вважає, що розфарбована фотографія виглядає гірше чорно-білої, бачачи невдалі приклади, і відмовляються від колоризації, і марно, адже якщо перетворити чорно біле фото в кольорове правильно, то воно стає таким, що не відрізняється від кольорового, і навіть набагато краще.

Які критерії можуть бути для перетворення монохромного зображення на кольорове:

- Реалістичність. Ідеально розфарбована фотографія виходить як звичайна кольорова. На ній не помітно слідів розмальовки та накладеного кольору.
- Природність. Особа має натуральний здоровий відтінок і виглядає живим, а не плоским або намальованим. Кілька осіб не виглядають однотипно, кожне має індивідуальністю.
- Обсяг. Правильно розфарбована фотографія виглядає завжди об'ємнішою ніж чорно-біла. Якщо об'єкти і особи стали більш плоскими, ніж на чорно-білому знімку, значить щось зроблено не так.
- Історична достовірність. Особливо важливо потрапити в потрібний відтінок, коли потрібно відновити кольори військової форми, нагород або зброї. На таких фотографіях колір потрібен не тільки для краси, він несе інформацію.
- Гармонія кольору. Основна мета розфарбовування фотографії – не формальна наявність якого-небудь кольору. Колір повинен робити фотографію виразніше і подобатися. Колір – це ціла наука, а не просте «прикрашення». Занадто помітні кольори і поєднання або навпаки тьмяні й сумні, неправильні поєднання – все це створює

загальне негативне враження. Коли колір підібраний правильно, фотографія викликає позитивні враження і емоції.

- Акуратність. Кольори знаходяться на своєму місці і не видають свого штучного походження, межі переходу кольорів не виділяються, вони не різкі і не розмиті.

Одним із важливих аспектів при колоризації зображення є покращення якості цього зображення загалом. У монохромних зображеннях часто бувають дефекти (такі як подряпини, затертості) або ж артефакти, які виникають при оцифруванні такого зображення. Ці аспекти негативно впливають на результат колоризації такого зображення. Тому перед початком роботи з чорно-білою картинкою потрібно позбутись цих дефектів. Як правило, на таких фотографіях присутні подряпини, що утворилися з часом в результаті неправильного зберігання, неакуратного поводження. Сучасні ж фотографії в основному зберігаються на електронних носіях і такої проблеми не виникає. Але також до дефектів такого роду фотографій як правило відносяться надмірна затемненность або навпаки освітленність, шуми, перегини і обірвані кути. Цифрова реставрація за допомогою спеціальних програм – це досить трудомісткий процес, що найчастіше вимагає багато часу. Частина роботи, яку доводиться робити дизайнеру, можна було б автоматизувати.

Завдання полягає в розгляді методів усунення подряпин при використанні методів аналітичної апроксимації та застосування нейронних мереж.

Об’єкт дослідження – каскад нейромереж для колоризації та обробки чорно-білого зображення.

Предмет дослідження – методи усунення дефектів на оцифрованих зображеннях, та методи розфарбування монохромного зображення.

1.2 Огляд існуючих рішень

1.2.1 «Colorize Photos»

«Colorize Photos» (рис.1.2)– онлайн сервіс на сайті Algoritmia. Сервіс використовує навчену на мільйоні зображень нейростіку для реконструювання чорно-білих зображень в режимі реального часу. Серед важливих функцій сервісу, є можливість порівняння вихідного зображення з початковим(рис.1.3).

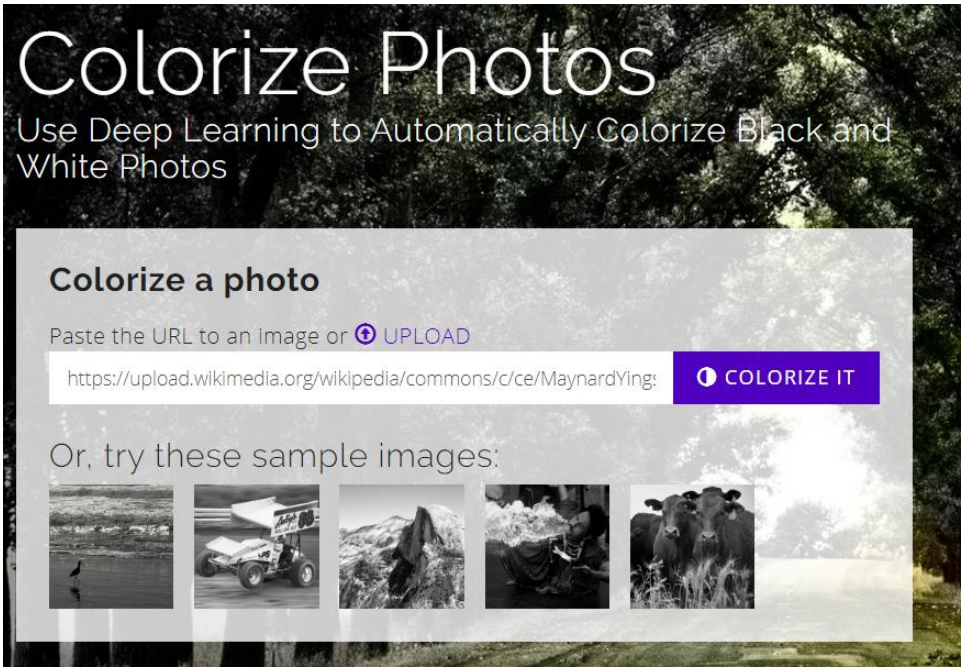


Рис. 1.2. Інтерфейс сервісу Colorize Photos

Рис. 1.3. Процес обробки зображення сервісом Colorize Photos

Із плюсів можна виділити:

– простота і інтуїтивність сервісу;
– можливість порівняння обробленого зображення з початковим в режимі реального часу(рис 1.3);
– можливість часткової обробки зображення;
– швидкісний алгоритм, що дозволяє проводити реконструкцію в режимі реального часу.

Основним недоліком даного сервісу є те, що через акцент на велику швідкість обробки зображення, результат обробки не завжди є задовільним, алгоритм погано реагує на можливі дефекти зображення і потребує усунення їх сторонніми програмами до процесу обробки. Також, можна побачити, що алгоритм видає перенасичені червоним кольором кольори, що негативно впливає на загальний результат обробки фотографії.

1.2.2 «Комп'ютерний зір від Mail.ru»

«Комп'ютерний зір від Mail.ru»– онлайн сервіс, що працює на технології Vision від Mail.Ru Group. Технологія Vision – це технологія розпізнавання осіб і об'єктів на базі машинного навчання. Даний сервіс дає доступ до багатьох

функцій розпізнавання та редагування зображення за допомогою нейросітки, одна з таких функцій є реконструювання чорно-білої фотографії в кольорову. Також серед представлених можливостей сервісу є пошук і реставрація дефектів зображення. Робота сервісу показана на фотографії нижче, зображення до обробки(рис.1.4), зображення після(рис.1.5) реставрації.

Рис. 1.4. Зображення над яким проводилось реконструювання кольору

Рис. 1.5. Отримане вихідне кольорове зображення після реконструювання

Серед плюсів виділяється:

– досить відома компанія;

– різноманітність функцій;
– можливість покращення зображення;
– реставрація дефектів;
– підібрані природні кольори.

Найбільшим мінусом даного аналогу є швидкість роботи алгоритмів. Використовуються складні алгоритми для покращення якості зображення, та реконструювання, що в свою чергу дає відчутну затримку при роботі, потрібно чекати декілька хвилин щоб сервіс опрацював одне зображення. Також, одним із мінусів є недоступність сервісу на просторі України.

1.2.3 «Колор»

«Колор»(рис.1.6) – мініатюрний сайт розроблений групою Artlebedev. Даний сайт має мінімалістичний інтефейс з тільки однією функцією – загрузки зображення для подальшої його оброки алгоритмами сайту. На відміну від інших аналогів цей сервіс є самодостатнім і не входить в загальний комплекс функцій на основі алгоритмів машинного навчання.

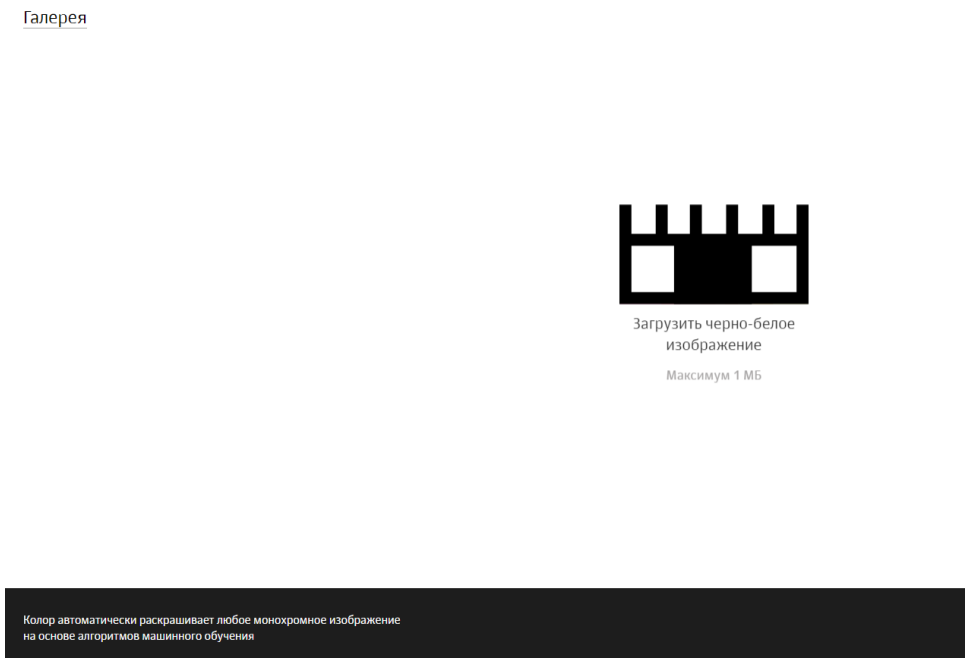


Рисунок 1.6. Інтерфейс сайту «Колор»

Рисунок 1.7. Результати оброки зображення сайтом «Колор»

Серед плюсів виділяються наступні:

– Простота інтерфейсу;
– Швидкість роботи алгоритму;
– Навчається в процесі обробки, можливо покращення результатів обробки фото після декількох прогонів сервісу;

Основним недоліком даного сервісу є досить незадовільні результати оброки зображення алгоритмом. Ситуація може покращитись при прогонці зображення через сервіс декілька разів, але це є незручним і не гарантує значного покращення вихідного зображення. Також цей сервіс додає до вихідного зображення свою «вотермарку». Ще одним важливим недоліком є те, що сервіс не приймає зображення розміром понад 1 мб.

Висновки до розділу

В розділі проведено постановку проблеми із зазначенням її актуальності на даний час. Також здійснено огляд предметної області, огляд підходів до вирішення проблеми та аналіз сучасних аналогів з метою виявлення їх сильних та слабких сторін.

РОЗДІЛ 2. ОГЛЯД І ПОРІВНЯННЯ СУЧАСНИХ ПІДХОДІВ ТА ІНСТРУМЕНТІВ

2.1 Опис предметної області

2.1.1 Історія нейронної мережі

Дослідження в області штучних нейронних мереж пережили кілька періодів активізації.

Перший період був в 1943 році, коли нейрофізіолог Уоррен Мак-Каллох і математик Уолтер Піттс написали статтю про роботу нейронів. Для опису роботи нейронів мозку, вони змоделювали просту НС, з використанням електричного кола [4]. Їх основна ідея полягала в тому, що будь-який зв'язок типу "вхід-вихід" може бути реалізована штучної (формальної) НС.

Другим етапом у розвитку НС стало винахід перцептрону в 1957 році Френк Розенблат. У 1958 р Френк Розенблат продемонстрував комп'ютерну модель електронного пристрою, названу їм перцептроном, а в 1960 р - перший діючий нейрокомп'ютер «Марк-1», який моделював спільну роботу людського ока і мозку [5]. Основним призначенням машини було розпізнавання. Кінцем цього етапу стала публікація Марвіна Лі Мінського і Сеймура Пайперт в 1969 році, в якій вони вказали на важливий клас задач, які одношаровий персептрон вирішувати не може. Мінський і Пайперт дискредитували дослідження НС і фінансування в області штучного інтелекту. Але незважаючи на демонстрацію Мінським і Пайперт обмежень перцептронів, дослідження нейронної мережі все ж тривали.

У 1982 році Джон Хопфилд - фізик зі світовим ім'ям, зацікавившись нейронними мережами, написав дві особливо Популярні статті про НС і провів численні лекції по всьому світу, ніж переконав сотні висококваліфікованих учених, математиків і технологів приєднатися до формування полю НС. Хопфилд показав, що високосвязная мережу нейронів із зворотними зв'язками може бути описана як динамічна система, що володіє "енергією". При

асоціативному виклику мережу, що стартує в довільному (випадковому) стані, сходиться до кінцевого стійкого стану з найменшою енергією. Новий підхід до опису НС із зворотними зв'язками виявився дуже плідним.

Подібний прорив стався і в зв'язку з багатошаровими мережами без зворотних зв'язків. Для навчання таких мереж був розроблений алгоритм зворотного поширення помилки.

З середини 80-х років теорія нейронних мереж отримала «технологічний імпульс», викликаний появою нових доступних і високопродуктивних персональних комп'ютерів.

Найзначніших досягнень в даному питанні досягла американська компанія IBM (англ. Internationa Business Machines). Перші результати досліджень були продемонстровані 14 листопада 2009 року. Компанія представила на суд громадськості успішно змодельований мозок кішки. Правда, слід зазначити, що тоді його робота була в 643 рази повільніше реального часу.

2.1.2 Штучний нейрон

Штучні нейронні мережі є відносно грубими електронними моделями, заснованими на нервовому структурі головного мозку. Фундаментальним обробляють елементом нейронної мережі є нейрон. Нейрон (нервова клітина) - це спеціальна біологічна клітина, яка обробляє інформацію. За даними оцінки, в мозку існує величезна кількість нейронів, кожен з яких володіє приблизно 1011 численними взаємозв'язками.

На рис. 2.1 представлена схема біологічного нейрона.

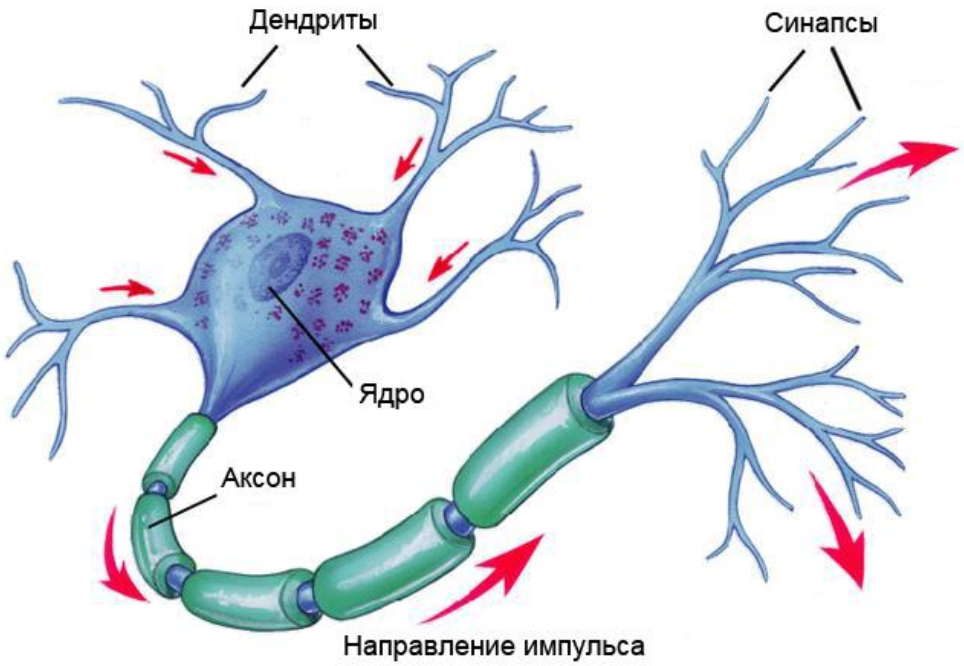


Рисунок 2.1 – Біологічний нейрон

Як показано на наведеній вище схемі, типовий нейрон складається з наступних чотирьох частин, за допомогою яких ми можемо пояснити його роботу:

- Дендрити – деревоподібні гілок, що відповідає за отримання інформації від інших нейронів, до яких підключений даний. В іншому сенсі, ми можемо сказати, що це «вуха» нейрона;
- Ядро – це тіло клітини нейрона, що відповідає за обробку інформації, отриманої від дендритів;
- Аксон – такий собі «кабель», за допомогою якого нейрони посилають інформацію;
- Синапси – це з'єднання між аксонів і дендритами інших нейронів.

Останні експериментальні дані надали додаткові докази того, що нейрони структурно складніше влаштовані, ніж спрощено описано вище. Вони значно складніші, ніж існуючі штучні нейрони, які вбудовані в сьогоднішні штучні нейронні мережі. У міру того, як біологія забезпечує краще розуміння нейронів, і в міру розвитку технологій, розробники мереж можуть продовжувати удосконалювати свої системи, спираючись на розуміння людського мозку.

Але в даний час грандіозне відтворення мозку – не є метою штучних нейронних мереж. Навпаки, нейромережеві дослідники шукають способи застосування можливостей природи, для яких люди можуть спроектувати рішення проблем, які не були вирішені за допомогою традиційних обчислень.

Для цього потрібна основна одиниця нейронних мереж – штучні нейрони, що моделюють чотири основні функції біологічних нейронів(рис.2.2).

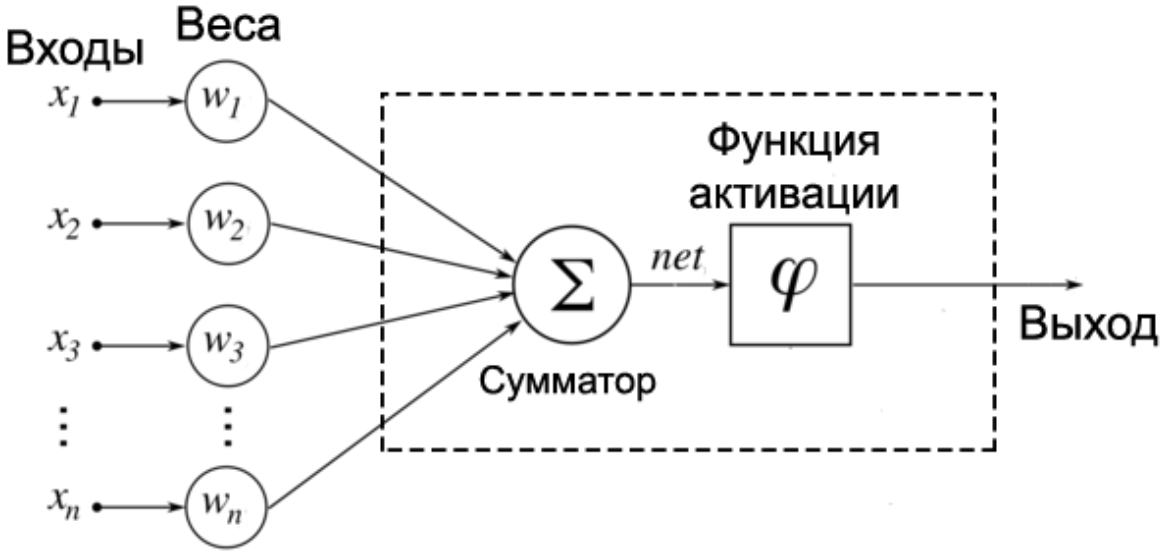


Рисунок 2.2 – Штучний нейрон

2.1.3 Перцептрон

Перцептрон є абстрактною моделлю біологічного нейрона. По суті, це дуже простий процесор. Перцептрон приймає кілька двійкових входів, *x1*, *x2*, ..., *xn* і виробляє один двійковий вихід. Раніше згаданий Розенблатт запропонував просте правило для обчислення вихідних даних. Він ввів ваги, *w1, w2, ..., wn* – речові числа, що виражають важливість відповідних вхідних даних для виведення. Про них ми вже згадували, коли говорили про завдання класифікації образів і вводили формулу (1). Вихід нейрона від того, чи є результат роботи суматора – сума $\sum_{j=1}^{n} \omega_j x_j$, звана зваженою, менше або більше деякого порогового значення. Так само, як і ваги, порогове значення є

дійсним числом, що є параметром нейрона. Наведемо це в більш точних алгебраїчних термінах, як у формулі (1):

$$output = \begin{cases} 0, \text{якщо } \sum_{j=1}^{n} \omega_j x_j \leq \text{порогове значення;} \\ 1, \text{якщо } \sum_{j=1}^{n} \omega_j x_j > \text{порогове значення.} \end{cases} \quad (1)$$

Це основна математична модель. Ви можете думати про перцептроном так: це пристрій, який приймає рішення, зважуючи різні види даних. Тому має бути очевидним, що складна мережа перцептронів може приймати досить складні рішення.

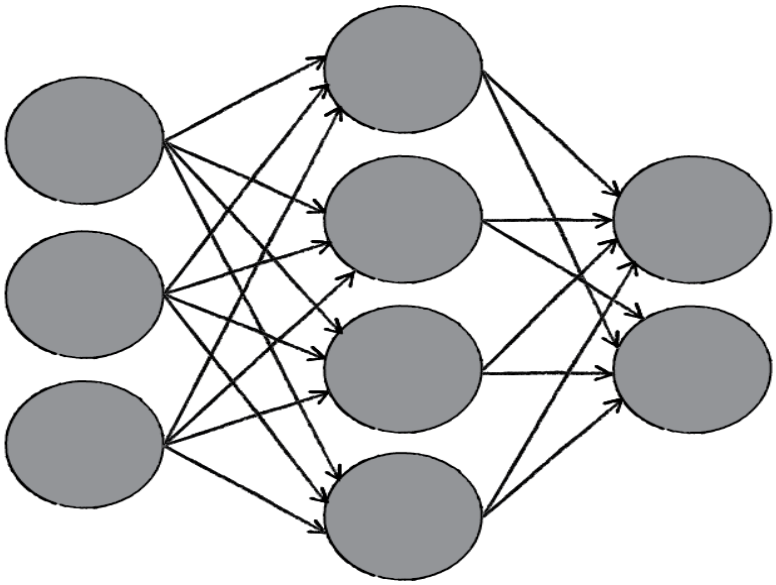


Рисунок 2.3 – Мережа перцептронів

Розглянемо мережу перцептронів. У цій мережі перша колонка перцептронів – то, що ми будемо називати вхідним шаром перцептронів – приймає прості рішення, зважуючи вхідні дані. Кожен з перцептронів другого шару приймає рішення, зважуючи вихідні дані перцептронів вхідного шару. Тобто, перцептрон в другому шарі може приймати рішення на більш складному і більш абстрактному рівні, ніж перцептрони в першому шарі. І ще більш складні рішення може приймати перцептрон в третьому шарі і т.д. Таким чином, багатошарова мережа перцептронів може брати участь в ухваленні складних рішень.

Раніше було сказано, що перцептрон має тільки один вихід. У мережі вище перцептрони виглядають так, як ніби вони мають кілька виходів. Насправді, вони все ще мають лише одним виходом. Стрілки з декількома вихідними даними – це просто корисний спосіб вказати, що вихідні дані перцептрону використовуються в якості вхідних даних для декількох інших перцептронів. Це менш громіздко, ніж малювати одну вихідну лінію, яка потім розділяється[4].

Ми можемо спростити спосіб опису перцептронів. Умова – $\sum_{j=1}^{n} \omega_j x_j \leq \text{порогове значення}$ – громіздко, і ми можемо зробити дві зміни, щоб спростити його. Перша зміна полягає в тому, щоб записати суму творів $\omega_j x_j$ у вигляді векторного твори, $\sum_{j=1}^{n} \omega_j x_j$, де w і x – вектори, компоненти яких є вагами і вхідними даними відповідно. Друга зміна полягає в тому, щоб перенести порогове значення на іншу сторону нерівності і замінити його число b, званим зміщенням перцептрону. Використовуючи вектора і зміщення замість граничного значення, формулу (1) можна переписати і представити у вигляді (2):

$$output = \begin{cases} 0, \text{якщо } w \cdot x + b \leq 0; \\ 1, \text{якщо } w \cdot x + b > 0. \end{cases} \quad (2)$$

Навчальні алгоритми автоматично налаштовують ваги і відхилення мережі нейронів у відповідь на зовнішній стимул, без прямої участі програміста. Процес навчання передбачає мале зміна ваг і відхилень, щоб отримати мале зміна на виході, наближаючись до кращого результату класифікації. Але мережу персептронов так не працює, і навіть мале вплив може викликати різкі зміни, приводячи до непередбачуваних наслідків. Для вирішення цієї проблеми використовують так звані функції активації. Для перцептрону функцією активації, як можна побачити з формул, є одинична функція.

2.1.4 Функції активації

Ми вже з'ясували, що нейрон повністю описується своїми вагами і функцією активації, що позначається $f(\psi)$, де ψ – зважена сума. Отримавши вектор x в якості вхідних даних, нейрон видає деяке число на виході.

Активаційна функція може бути різного виду. Найбільш широко використовуються варіанти наведені в таблиці (за s позначена зважена сума):

Таблиця 2 – Перелік функцій активації нейронів

Назва	Формула	Область значень
Порогова	$f(s) = \begin{cases} 0, s < 0 \\ 1, s \geq 0 \end{cases};$	*(0,1)*
Знакова	$f(s) = \begin{cases} 1, s > 0 \\ -1, s \leq 0 \end{cases};$	*(-1,1)*
Сигмоїдальна	$f(s) = \frac{1}{1+e^{-s}};$	*(0,1)*
Напівлінійна	$f(s) = \begin{cases} 0, s \leq 0; \\ s, s > 0; \end{cases}$	*(0; ∞)*
Лінійна	$f(s) = s;$	*(-∞; +∞)*
Радіальна базисна (гауссова)	$f(s) = exp(-s^2);$	*(0,1)*
Напівлінійна з насиченням	$f(s) = \begin{cases} 0, s \leq 0; \\ s, 0 < s < 1; \\ 1, s \geq 1; \end{cases}$	*(0,1)*
Лінійна з насиченням	$f(s) = \begin{cases} -1, s \leq -1; \\ s, -1 < s < 1; \\ 1, s \geq 1; \end{cases}$	*(-1,1)*
Гіперболічний тангенс	$f(s) = \frac{e^3 - e^{-3}}{e^3 + e^{-3}};$	*(-1,1)*
Трикутна	$f(s) = \begin{cases} 1 - \lvert s \rvert, \lvert s \rvert \leq 1; \\ 0, \lvert s \rvert > 1. \end{cases}$	*(0,1)*

Однією з найбільш поширених є нелінійна функція з насиченням, так звана логістична функція або сигмоїдальна, представлена формулою (3):

$$f(s) = \frac{1}{1+e^{-s}}. \quad (3)$$

Логістична функція має такі властивості:

- вона є «стискаючою» функцією, тобто незалежно від аргументу (зваженої суми), вихідний сигнал завжди буде в межах від 0 до 1;
- у всіх точках вона має похідну, і ця похідна може бути виражена через цю ж функцію. Похідна представлена формулою (4):

$$f'(s) = f(s)(1-f(s)). \quad (4)$$

2.1.5 Архітектура нейронних мереж

Існує безліч видів нейронних мереж.

Мережа прямого поширення сигналу (мережа прямої передачі) – нейронна мережа без зворотних зв'язків. У цій мережі поширення сигналу однонаправлено, тобто немає зворотного зв'язку. Від вхідного шару сигнал обробляється шар за шаром в напрямку виходу. Через відоме число кроків на вихідному шарі з'являється відповідь мережі.

Мережі прямого поширення є добре вивченими і відносно простими в реалізації. Їх недоліком є необхідність великого числа нейронів для виконання складних завдань.

Ланцюги Маркова – свого роду попередники машин Больцмана і мереж Хопфілда, про які сказано нижче. У ланцюгах Маркова ми задаємо ймовірності переходу з поточного стану в сусідні. Крім того, це ланцюга не мають пам'яті: подальший стан залежить тільки від поточного і не залежить від всіх минулих станів. Хоча ланцюг Маркова не можна назвати нейронною мережею, вона близька до них і формує теоретичну основу для машин Больцмана і мереж Хопфілда.

Мережа Хопфілда (модель Хопфілда), є одним з видів мереж асоціативної пам'яті. Це одношарова нейронна мережа, в якій кожен нейрон пов'язаний з усіма іншими, має по одному входу і виходу. Жорстка функція активації генерує два значення: -1 (загальмований) і +1 (збуджений). У моделі використовується принцип зберігання інформації як динамічно стійких

атракторів. Енергетична функція зменшується в процесі навчання поки не досягає локального мінімуму (атрактора), в якому зберігає постійне значення [7].

Машини Больцмана багато в чому схожі на мережі Хопфілда, але в них деякі нейрони позначені як вхідні, а деякі залишаються прихованими. Вхідні нейрони стають вихідними, коли все нейрони в мережі оновлюють свої статки. Спочатку вагові коефіцієнти присвоюються випадковим чином, потім відбувається навчання методом зворотного поширення, або останнім часом все частіше за допомогою алгоритму contrastive divergence (коли градієнт обчислюється за допомогою марковської ланцюга). Машини Больцмана - стохастична нейронна мережа, так як в навчанні задіяна ланцюг Маркова.

Рекурентні мережі – це глибокі мережі, в яких присутні зворотні зв'язки. Це означає, що є присутнім хоча б один шар, сигнали з якого надходять на нього ж, або на один з попередніх шарів. Нейрони беруть участь в обробці інформації багаторазово, що дозволяє використовувати динамічні властивості мережі. Такі мережі дозволяють скоротити число нейронів. На основі рекурентних мереж розроблені різні моделі асоціативної пам'яті. Особливо ці мережі стали в нагоді в області розпізнавання мови[4].

Мережа Хемінга (Класифікатор по мінімуму відстані Хеммінга) – інше приклад нейронної мережі асоціативної пам'яті. Принцип роботи заснований на обчисленні відстані Хеммінга від вхідного вектора до всіх векторів-зразків, відомих мережі. При надходженні вхідного образу, мережа вибирає зразок з найменшим до нього відстанню Хеммінга і відповідний йому вихід активізується[4].

Мережа Кохонена – одношарова мережу з настроюються ваг. У той час як один нейрон збуджується, всі інші виходи шару придушуються. Ваги підлаштовуються так, що вхідні образи з одного класу активують один і той же вихідний нейрон. Так вхідні вектора класифікуються за схожим групам. Це відображає одне з найважливіших властивостей мережі Кохонена – здатність

до узагальнення. Вектор кожного з нейронів мережі замінює групу відповідних йому класифікуються векторів.

Глибокі мережі довіри – це мережі, що представляють собою каскад Обмежених Машин Больцману. Стандартна Машина Больцмана складається з повнозв'язних "видимих" і "прихованих" нейронів, які беруть бінарні значення, певні векторами. Обмежені машини відрізняються тим, що нейрони одного класу не пов'язані між собою. Ці мережі цікаві тим, що можуть грати роль генеруючих моделей. Іншими словами, мережа навчена розпізнавати, наприклад, рукописний текст в теорії може бути використана для генерації зображень, які виглядають як рукописний текст.

Згортаючи нейронні мережі і глибокі згортаючи нейронні мережі кардинально відрізняються від інших мереж. Вони використовуються в основному для обробки зображень, іноді для аудіо та інших видів вхідних даних. Типовим способом їх застосування є класифікація зображень. Такі мережі зазвичай використовують «сканер», що не обробляє всі дані за один раз.

2.1.6 Архітектура нейронних мереж

Кожна нейронна мережа вимагає навчання, в іншому випадку правильний результат навряд чи буде отриманий. Навчання нейронної мережі (Training) – це пошук такого набору вагових коефіцієнтів, при якому вхідний сигнал після проходу по мережі перетворюється в потрібний нам вихідний.

Розглянемо деякі з методів навчання нейронних мереж.

1) Метод зворотного поширення помилки.

Цей метод є основним і має ще одну назву – Backpropagation, так як використовує алгоритм градієнтного спуску. Тобто за допомогою руху уздовж градієнта розраховується локальний мінімум і максимум функції.

У процесі надходження інформації нейронна мережа послідовно передає її від одного нейрона до іншого за допомогою синапсів, до того моменту, поки

інформація не опиниться на вихідному шарі і не буде видана як результат. Такий спосіб називається передачею вперед. Після того, як результат отриманий обчислюється помилка і на її снованіі виконуємо зворотний передачу. Суть, якої – послідовно змінити вагу синапсів починаючи з вихідного і просуваючись до вхідного шару. При цьому значення ваги змінюється в бік кращого результату. Для використання такого методу навчання підійдуть тільки ті функції активації, які можна диференціювати. Так як зворотне поширення обчислюється за допомогою вираховування різниці результатів і множення його на похідну функції від вхідного значення. Для того, щоб успішно провести навчання, необхідно поширити отриману помилку на весь вага мережі. Вирахувавши помилку на вихідному рівні, а також там можна обчислити дельту, яка буде послідовно передаватися між нейронами. Потім необхідно провести розрахунок градієнта для кожної вихідного зв'язку. Потім маючи всі необхідні дані потрібно оновити програмне забезпечення ваг і розрахувати завдяки функції методу значення, яке стане величиною зміни. При цьому не варто забувати про момент і швидкість навчання [8].

Одна ітерація цього методу дає невеликий відсоток зменшення помилки, тому повторювати їх необхідно знову і знову поки показник помилки не буде наближений до 0.

2) Метод пружного поширення

Попередній спосіб навчання, представлений вище, має недолік у вигляді великих тимчасових витрат на процес навчання недоречних в разі необхідності отримати швидкий результат. Для прискорення процесу було запропоновано чимало додаткових алгоритмів, що прискорюють процес. Одним з яких і є поточний метод. Справжній алгоритм використовує в якості основи навчання по епохах і застосовує тільки знаки похідних окремого випадку для коригування вагових коефіцієнтів. Використовується певний правило, за яким здійснюється розрахунок величини корекції вагового коефіцієнта:

Якщо на цьому етапі розрахунків похідна змінює свій знак, значить, зміна була надто великою і локальний мінімум був упущений і потрібно зробити відкат, тобто вага повернути в зворотну позицію, а величину зміни зменшити.

Якщо знак похідної не змінився, то величина зміни ваги, навпаки, збільшується для більшої збіжності.

Якщо основні параметри корекції ваги зафіксувати, то настройки глобальних параметрів можна уникнути. І це стане ще однією перевагою поточного методу над попереднім. Для цих параметрів є рекомендовані значення, однак, ніяких обмежень на їх вибір не накладається.

Щоб вага не приймав занадто великі або малі значення використовуються встановлені обмеження величини корекції. Величина компенсації також обчислюється за певним правилом:

Тобто якщо похідна функції в конкретній точці змінює знак з плюса на мінус, означає, що помилка зростає і вага вимагає корекції і відбувається його зменшення, в іншому випадку - збільшення.

Такий підхід дозволяє домогтися збіжності нейромережі швидше в кілька разів на відміну від попереднього варіанту навчання.

3) Генетичний алгоритм

Третій найбільш цікавий алгоритм навчання штучних нейронних мереж – Genetic Algorithm. Він являє собою спрощену інтерпретацію природного алгоритму, заснованого на схрещуванні результатів. Тобто, по суті, відбувається схрещування результатів, вибір найкращих і формування на їх основі нового покоління.

У разі якщо результат не влаштовує алгоритм повторюється поки покоління не ставати ідеальним. Алгоритм може завершитися без досягнення потрібного результату якщо кількість спроб буде вичерпано або ж буде вичерпано час на мутацію. Цей алгоритм застосуємо до процесу оптимізації ваги нейронної мережі, при заданої за замовчуванням топології.

При цьому вага кодується двійковим кодом і кожен результат визначається повним набором ваги. Оцінка якості відбувається методом обчислення помилки на виході.

2.2 Огляд платформи .Net

.NET Framework – це технологія, яка підтримує створення і виконання нового покоління додатків і веб-служб XML. При розробці платформи .NET Framework враховувалися наступні цілі:

- Забезпечення узгодженого об'єктно-орієнтованого середовища програмування для локального збереження і виконання об'єктного коду, для локального виконання коду, розподіленого в Інтернеті, або для віддаленого виконання.
- Забезпечення середовища виконання коду, що мінімізує конфлікти при розгортанні програмного забезпечення та управлінні версіями.
- Забезпечення середовища виконання коду, що гарантує безпечне виконання коду, включаючи код, створений невідомим або не повністю довіреною стороннього постачальника.
- Забезпечення середовища виконання коду, що виключає проблеми з продуктивністю середовищ виконання сценаріїв або інтерпретується коду.
- Забезпечення єдиних принципів розробки для різних типів додатків, таких як додатки Windows і веб-додатки.
- Взаємодія на основі промислових стандартів, яке гарантує інтеграцію коду платформи .NET Framework з будь-яким іншим кодом.

Платформа .NET Framework складається з загальномовного середовища виконання (середовища CLR) і бібліотеки класів .NET Framework. Основою платформи .NET Framework є середовище CLR. Середовище виконання можна

вважати агентом, який керує кодом під час виконання і надає основні служби, такі як управління пам'яттю, управління потоками і віддалене взаємодія. При цьому середовищем накладаються умови суворої типізації та інші види перевірки точності коду, що забезпечують безпеку і надійність. Фактично основним завданням середовища виконання є управління кодом. Код, який звертається до середовища виконання, називають керованим кодом, а код, який не звертається до середовища виконання, називають некерованим кодом. Бібліотека класів є комплексною об'єктно-орієнтованої колекцією повторно використовуваних типів, які застосовуються для розробки додатків – починаючи з звичайних додатків, що запускаються з командного рядка, і додатків з графічним інтерфейсом (GUI) і закінчуючи додатками, що використовують останні технологічні можливості ASP.NET, такі як веб-форми і веб-служби XML[6].

Для розробки веб сервісу використовується платформа ASP.NET, що на даний момент має дві потужні реалізації ASP.NET MVC та ASP.NET Core.

2.1.1 Платформа ASP.NET MVC

Платформа ASP.NET MVC являє собою фреймворк для створення сайтів і веб-додатків за допомогою реалізації паттерна MVC:

- Концепція паттерна (шаблону) MVC (model - view - controller) передбачає поділ додатка на три компоненти:
- Контролер (controller) представляє клас, що забезпечує зв'язок між користувачем і системою, уявленням і сховищем даних. Він отримує введені користувачем дані і обробляє їх. І в залежності від результатів обробки відправляє користувачеві певний висновок, наприклад, у вигляді уявлення.
- Уявлення (view) – це власне візуальна частина або призначений для користувача інтерфейс програми. Як правило, html-сторінка, яку користувач бачить, зайшовши на сайт.
- Модель (model) представляє клас, що описує логіку використовуваних даних.

Рисунок 2.2 – Патерн MVC в ASP.NET

У цій схемі модель є незалежним компонентом – будь-які зміни контролера або уявлення не зачіпають модель. Контролер і уявлення є відносно незалежними компонентами, і нерідко їх можна змінювати незалежно один від одного [4].

Завдяки цьому реалізується концепція поділ відповідальності, в зв'язку з чим легше побудувати роботу над окремими компонентами. Крім того, внаслідок цього додаток має кращу тестуємість. І якщо нам, припустимо, важлива візуальна частина або фронтенд, то ми можемо тестувати уявлення незалежно від контролера. Або ми можемо зосередитися на бекенда і тестувати контролер.

Конкретні реалізації та визначення даного патерну можуть відрізнятися, але в силу своєї гнучкості і простоти він став дуже популярним останнім часом, особливо в сфері веб-розробки.

2.1.2 Платформа ASP.NET Core

ASP.NET Core – це кроссплатформенная середа з відкритим кодом для створення сучасних хмарних веб-додатків в Windows, macOS або Linux.

ASP.NET Core надає наступні переваги:

- Єдине рішення для створення призначеного для користувача веб-інтерфейсу і веб-API.
- Розроблено для тестованості.
- Razor Pages робить створення кодів сценаріїв для сторінок простіше і ефективніше.

- Blazor дозволяє використовувати в браузері мову C # разом з JavaScript. спільне використання серверної і клієнтської логік додатків, написаних за допомогою .NET;
- Можливість розробки і запуску в ОС Windows, macOS і Linux.
- Відкритий вихідний код і орієнтація на співтовариство.
- Інтеграція сучасних клієнтських платформ і робочих процесів розробки.
- Підтримка розміщення служб віддаленого виклику процедур (RPC) за допомогою gRPC.
- Хмарна система конфігурації на основі середовища.
- Вбудоване введення залежностей.
- Спрощений високопродуктивний модульний конвеєр HTTP-запитів.
- Наступні можливості розміщення:
- Kestrel
- служби IIS
- HTTP.sys
- Nginx
- Apache
- Docker
- Управління паралельними версіями.
- Інструментарій, що спрощує процес сучасної веб-розробки.

2.3 Огляд алгоритмів машинного навчання

Розглянемо деякі алгоритми які ефективно вирішують задачу обробки вхідного зображення.

2.3.1 Алгоритм класифікації пікселів

Для класифікації пікселів використовується оптичний поріг. Оптичний потік – технологія, що використовується в різних областях комп'ютерного зору для визначення зрушень, сегментації, виділення об'єктів, стисненні відео. Оптичний потік – зображення видимого руху, що представляє собою зсув кожної точки між двома зображеннями. Оптичний потік використовується при визначенні руху, і існують методи, що обчислюють рух між двома кадрами, узятими в момент часу t і t1, в кожному пікселі. Алгоритми на його основі застосовуються при вирішенні завдань, пов'язаних з кодуванням рухів і розробці систем стереозрения. За допомогою алгоритмів, що використовують оптичний потік можна визначити не тільки рух об'єктів, а й створити тривимірну структуру сцени.

2.3.2 Алгоритм Хафа

Стандартний алгоритм складається з наступних кроків:

- Вибір кроку дискретизації;
- Заповнення матриці;
- Аналіз матриці;
- Виділення кривих.

На першому кроці здійснюється вибір сітки дискретизації. При виборі сітки важливо знайти оптимальний розмір, так як у випадку занадто великий сітки є ймовірність попадання точок, які лежать на різних кривих. Якщо вибрати дуже дрібну, то існує ймовірність розмиття максимумом шуканої кривої, так як точки однієї кривої можуть потрапити в різні осередки.

Другий крок є найбільш трудомістким етапом в алгоритмі, складність якого залежить розмірності простору і частоти дискретизації. Кількість осередків зростає зі збільшенням розмірності і зменшенням сітки.

Існує кілька модифікацій стандартного алгоритму Хафа:

Комбінаторні перетворення Хафа– цей метод створювався для виявлення прямих ліній на зображенні. При такій постановці завдання ми працюємо з

плоским бінарним зображенням. Вважаємо, що цікавлять нас точки мають один колір, фон має інший. Відмінності від стандартного методу в тому, щоб розбити зображення на маленькі ділянки. Далі на кожному з цих ділянок визначаються параметри точок, через які проходить пряма. Після цього, параметри заносяться в якусь комірку і лічильник збільшується. Таким чином відбувається скорочення кількості переборовши. У разі малого числа точок, що потрапляють в ділянку, і зменшення сітки дискретизації число записів скорочується.

Ієрархічне перетворення Хафа – інший метод пошуку ліній на бінарному зображенні:

- Початкове зображення розбивається регулярною сіткою
- Використовуючи перетворення Хафа, в кожному фрагменті оброблюваного зображення виявляються прямі,
- Проводиться ієрархічне злиття

На кожному етапі алгоритму ми розглядаємо чотири сусідніх фрагмента. На кожній дільниці виділяються лінії, які потім об'єднуються, використовуючи перетворення Хафа, і тим самим об'єднуються фрагменти. У разі, якщо лінія не зливається з лініями з сусідніх ділянок, то вона більш не розглядається. Подальше злиття відбувається до отримання загального зображення. Таким чином знижується складність виконання, так як при розбитті можна використовувати більш велику сітку. Недоліком є те, що лінія, що має велику довжину, може представлятися декількома лініями, в результаті того, що кінці лінії на низьких рівнях можуть не бути колінеарними.

Адаптивне перетворення Хафа – цей метод створено для зменшення місця, необхідного для зберігання матриці і збільшення продуктивності пошуку кривих ліній. При роботі алгоритму використовується матриця малого розміру. Етапи роботи:

- Вибір матриці маленького розміру;
- Розмір осередку матриці зменшується з кожної ітерації, до тих пір, поки не буде досягнутий заданий розмір;

- Заповнення матриці відбувається стандартним способом;
- Знаходження осередки з максимальною величиною лічильника, після чого цей осередок вважається новим фазовим простором.
- Виділення кривої.

Основними перевагами є зменшена складність і сітка дискретизації, яка уточнюється на кожному етапі. Недоліком цього методу є те, що у випадку з кількома кривими з одного сімейства виникає необхідність повторення всього алгоритму для кожної з них.

2.4 Огляд мов програмування

У наш час існує багато мов програмування, як спеціальних так і загальних, які можна використовувати для вирішення задач машинного навчання. Проаналізувавши цей графік можна зробити декілька висновків. По-перше: у цій сфері використовується досить багато різних мов програмування. По-друге: популярність машинного навчання та науки про дані різко зросла за останні декілька років. Втретє: Python є лідером, слідом за ним йдуть R, Java та Scala. В останнє: ріст популярності Scala значно зріс. Так 4 роки назад Scala практично не використовувалася для машинного навчання, а зараз вона займає 4 місце. Це може бути викликано зростанням популярності платформи Apache Spark. У даній роботі використовується Python. Порівняємо R, Python та Scala для вирішення задач машинного навчання. Таблиця 2.1 показує кількість запитань на Stack Overflow, кількість пакетів в головному репозиторії та індекс Tiobe (чим менше тим краще). Індекс Tiobe – індекс, що оцінює популярність мов програмування використовуючи дані пошукових запитів, які включають у себе назву мови. Для формування цього індексу використовують такі сервіси, як: Google, Wikipedia, Youtube, Amazon, Bing, Yahoo. Розрахунок цього індексу проводиться раз у місяць. У випадку Scala треба порахувати кількість Java пакетів, так як Scala, як і Java інтерпритується у байт код та виконується JVM. Порахувати кількість Java пакетів не є об'єктивно можливим.

Таблиця 2.1

	Python	R	Scala
Запитання на Stack Overflow	527,550	122,907	46,580
Кількість пакетів	73,402 packages on PyPI	7,798 packages on CRAN	Java пакети
Індекс Tiobe	4	16	30

Порівнюючи різні мови треба не тільки акцентувати увагу на швидкості та можливостях, але й на суспільство розробників, які можуть допомагати тим хто починає вивчати нову мову та продовжувати розвиток та ріст різних існуючих мов.

2.5 Огляд бібліотек машинного навчання

2.5.1 Tensorflow

Tensorflow – бібліотека для побудови нейронних мереж. Вона орієнтована на представлення процесу обрахунку як направленого графу. Це зручно для обчислення великих задач. Tensorflow написаний на мові C++, також ця бібліотека надає можливість викликати ці функції на мові Python, тому швидкість обробки не зменшується. Також Tensorflow підтримує CUDA, програмно-апаратну архітектуру пралаельних обичлень, яка дозволяє збільшити обчислювальну продуктивнчть завдяки використанню графічних процесорів NVidia[5].

2.5.2 Scikit-learn

Scikit-learn – одна з найпопулярніших бібліотек для машинного навчання. Вона має дуже великий набір інструментів для аналізу даних. Вона побудована використовуючи бібліотеки орієнтовані на наукові дослідження,

такі як NumPy, SciPy та matplotlib. Серед усіх бібліотек машинного навчання scikit-learn має найбільш широкий набір алгоритмів та підходів. Вона слугує як фундамент для інших розробок[2].

2.5.3 Caffe

Caffe – бібліотека для вирішення задач візуального розпізнавання. Переважно вона використовується для створення глибоких нейронних мереж. Також ця бібліотека добре зінтегрована з GPU, що дозволяє значно прискорити обчилення задач, які добре виконуються на графічному процесорі. Caffe найчастіше знаходить застосування у наукових роботах[2].

2.5.4 Pyevolve

Pyevolve – бібліотека яка використовує генетичні алгоритми. Вона тестує нейрону мережу на деяких даних та використовує генетичні алгоритми для покращення якості моделі[5].

2.5.5 MLilb

MLlib – бібліотека для машинного навчання, яка використовується платформою Spark, яка використовується для реалізації розподілених систем. Вона орієнтована на масштабованість та паралельність, тому в ній реалізовані ті алгоритми, які можуть виконуватися паралельно. Використовувати MLlib можна на тих мовах, на яких можливо програмувати для платформи Apache Spark, а саме: Python, Scala та Java. MLlib використовує пакет Breeze, написаний на Scala. Він являє собою реалізації алгоритмів машинного навчання на чисельних методів[3].

2.6 Огляд бібліотек для обробки зображення

Для обробки зображення порівняно 2 популярні бібліотеки: opencv та scikit-image. Так як для бінаризації зображення використовується scikit-learn , то треба порівняти швидкість операції морфологічного відкриття для того щоб обрати кращу бібліотеку. В таблиці 2.2 приведені результати швидкості обрахування морфологічного відкриття з радіусом структурного елемента рівним 12 для 20 різних зображень. Як видно бібліотека opencv працює швидше у 450 разів, тому саме вона буде використана у роботі. Така велика різниця пояснюється тим, що бібліотека opencv використовує процедури на мові С, яка компілюється у машинний код, тоді як Python використовує інтерпретатор.

Таблиця 2.2

	OpenCv	Sckit-mage
Середній час обрахування морфологічного відкриття с радіусом структурного елемента 12, с	0.00385	1.73

Висновки до розділу

У даному розділі було виконано огляд відомих підходів до вирішення поставленного завдання. Для колоризації та покращення якості зображення використовують різні алгоритми машинного навчання (ймовірнісна нейронна мережа, метод головних компонент), та різні ознаки. Всі підходи показали гарні результат.

Також виконано огляд мов програмування та бібліотек, що використовуються для обробки зображення та машинного навчання. Проведено огляд алгоритмів.

РОЗДІЛ 3. РОЗРОБКА АЛГОРИТМІЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Усунення дефектів та покращення якості зображення

3.1.1 Методи усунення шуму

Залежно від розглянутої моделі шуму складність вирішення такого завдання може істотно змінюватися. У практичних завданнях, як правило, має сенс використовувати моделі імпульсного і адитивного гауссового шуму, так як вони близькі до найбільш поширених в реальному світі шумів.

Адитивний шум Гауса заснований на використанні нормально розподіленої випадкової величини з нульовим математичним очікуванням, значення якої додаються до кожного пікселя зображення. Ця модель описує шум, який природним чином виникає при захопленні зображення цифровими сенсорами, і для якого існують добре вивчені способи шумозаглушення - високу ефективність мають класичні лінійні фільтри, наприклад фільтр Вінера [3] однак, разом з шумом фільтрація впливає також на дрібні деталі, і в контексті даної задачі може виникнути необхідність використання нелінійних методів, таких як алгоритми анізотропної дифузії, білатеральні і трілатеральні фільтри[4]. Перераховані вище методи базуються на локалізованої оцінці градієнта зображення, наявності контурів і дрібних деталей, що в подальшому дозволяє послабити згладжування цих ділянок і зберегти більшу кількість деталей.

Імпульсний шум виражається в неправильному (фіксованому або випадковому) значенні частини пікселів зображення. Як правило, подібний шум виникає через помилки при передачі інформації. Для такої моделі ефективними є ранжуючі фільтри [5], метою яких є виявлення імпульсної помилки і коригування її з використанням оцінки на основі наявних даних, при цьому неушкоджені пікселі залишаються незмінними.

У якості більш загального рішення, що підходить для придушення як гаусових, так і імпульсних шумів, використовуються адаптивні алгоритми, що використовує деяку околицю пікселя для визначення виду і коригування шуму, властивого центру цієї околиці.

3.1.2 Детектування дефектів

Знаходження подряпин сам по собі процес складний і існує безліч підходів. У більшості випадків цей процес здійснюється шляхом ручної обробки, який вимагає навичок і часу. Існує також напівавтоматичний спосіб видалення дефектів. Для відновлення пошкодженого зображення користувачем необхідно виділити передбачувані дефекти для подальшої реконструкції в автоматичному режимі.

Так як нашим головним завданням не є детектування, то для простоти роботи будемо використовувати ручну класифікацію.

Існують так звані методи інтерполяції (inpainting) [10], [11], що дозволяють видаляти дефектні ділянки, за рахунок аналізу відомих областей зображення. Це процес відновлення втрачених або пошкоджених частин зображень і відео. У цифровому світі інтерполяція (інтерполяція зображення або інтерполяція відео) відноситься до застосування складних алгоритмів для заміни втрачених або пошкоджених частин даних зображення (в основному невеликих областей або для усунення невеликих дефектів).

Алгоритми інтерполяції зображень можна розділити на різні категорії:

1) Методи на основі синтезу текстур (texture synthesis based image inpainting) / пошуку схожих блоків (Exemplar-based method, EBM)

2) Методи на основі рішення диференціальних рівнянь в похідних (PDE)

3) Гібридні методи.

3.1.2 Методи автоматичного детектування

На даний момент існує велика кількість методів автоматичного детектування подряпин на зображенні. Найперші і найбільш прості в реалізації алгоритми відносяться до класу порогової обробки [2]. Суть даних методів полягає у виборі порогових значень, які найбільш максимально поділяють пікселі належать дефектів, від пікселів належить об'єктам на гістограмі яскравості зображення. Іншим різновидом порогових детекторів є методи, засновані на порогової сегментації зображень [3]. Основним недоліком методів порогової обробки є залежність коректності виявлення дефектів від апріорно заданих порогових значень.

Іншим класом детекторів є морфологічна обробка зображення [4]. Для виявлення подряпин на фото документах використовуються морфологічні операції, такі як перетворення Top Hat для виявлення світлих і темних тріщин відповідно. Також можливе використання різних комбінацій морфологічних операцій, таких як відкриття і закриття, ерозія і дилатація з попередньо обраним структуроутворюючих елементом. Даний клас детекторів має в своїй результуючій масці менше помилкових спрацьовувань, ніж детектори, засновані на пороговій обробці за умови ефективних методів попередньої обробки зображення і постобробки маски. З недоліків слід зазначити залежність ефективності роботи від апріорно заданих параметрів.

Ще одним класом способів є методи частотної обробки зображень [5,6]. Дані методи дозволяють виділяти текстурні особливості на зображенні схожі з базовою функцією.

До останньої групи детекторів слід віднести методи, засновані на навчанні алгоритму [7]. Суть методів навчання полягає у виборі дескрипторів, які найбільш точно характеризують передбачуваний дефект. Ефективність виявлення таких детекторів залежить від якості вибору дескрипторів [8]. Дана група детекторів дозволяє застосовувати кілька методів виявлення дефектів і дає інтегральну оцінку для пікселів або області зображення.

Зображення розглядається як спрощена математична модель, яка представляє собою двовимірну дискретну послідовність $Y_{i,j}, i=1,N, j=1,M$, виду $Y_{i,j}=(1-d_{i,j})\cdot S_{i,j}+d_{i,j}\cdot c_{i,j}$, де Y – спостережуване зображення, S – оригінальне (неушкоджене) зображення, d бінарна маска області з перекрученими значеннями (1 – відповідає спотвореним пікселям, 0 – відповідає не спотвореним пікселям), c – спотворені значення пікселів.

На рис. 3.1 наведена блок-схема запропонованого алгоритму виявлення подряпин на зображенні в автоматичному режимі. Вона складається з двох етапів: попередньої обробки і етапу виявлення дефектів з використанням локальних бінарних дескрипторів.

Рисунок 3.1 – Блок-схема алгоритму

Попередня обробка є важливим кроком при виявленні дефектів, призначена для придушення шумовий складової, а також усунення не великих дефектів. У даній роботі використовується алгоритм фільтрації BM3D [9]. На

рис. 3.2 показаний приклад попередньої обробки - а) вихідне зображення, б) відфільтроване зображення.

Рисунок 3.2 – Попередня обробка зображення

Для виявлення подряпин як текстурного дескриптора для локальних областей на зображенні використовуються локальні бінарні околиці (LBP). За допомогою методу опорних векторів (SVM), всі фрагменти зображення класифікуються на два типи – область з дефектом і область без дефекту [10]. Дескриптори LBP дозволяють отримати гістограму, яка описує текстурні і структурні особливості зображення. Така гістограма також є інваріантною до повороту і зміни рівня яскравості. Дефект є, як правило, невеликою, по відношенню до фонової області, лінією, яка порушує однорідність текстури фону.

Модифікований LBP оператор може бути записаний у такий спосіб(5):

$$LBP_{P,K} = \begin{cases} \sum_{p=1}^{P} f(g_p - g_0) & if\ U \le U_T \\ P+1 & otherwise \end{cases}, \quad f(x) = \begin{cases} 1, x \ge 0 \\ 0, x < 0 \end{cases}, \tag{5}$$

де P число сусідів, K радіус, U число переходів між 1 і 0.

Таким чином, формується гістограма для локальної області, що представляє собою дескриптор для аналізу локальних областей в зображення на наявність подряпин (рис. 3.3) [11,12].

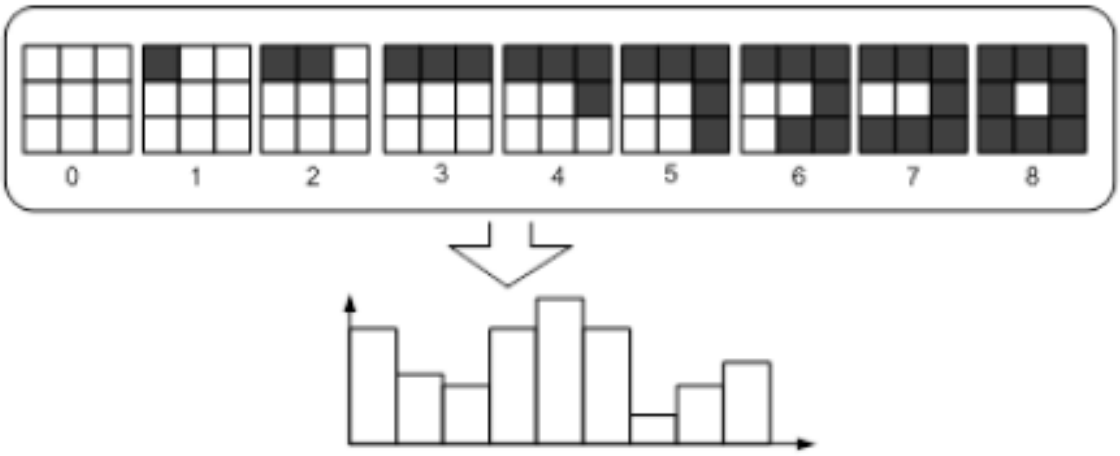


Рисунок 3.3 – Оператор і гістограма для однорідних околиць

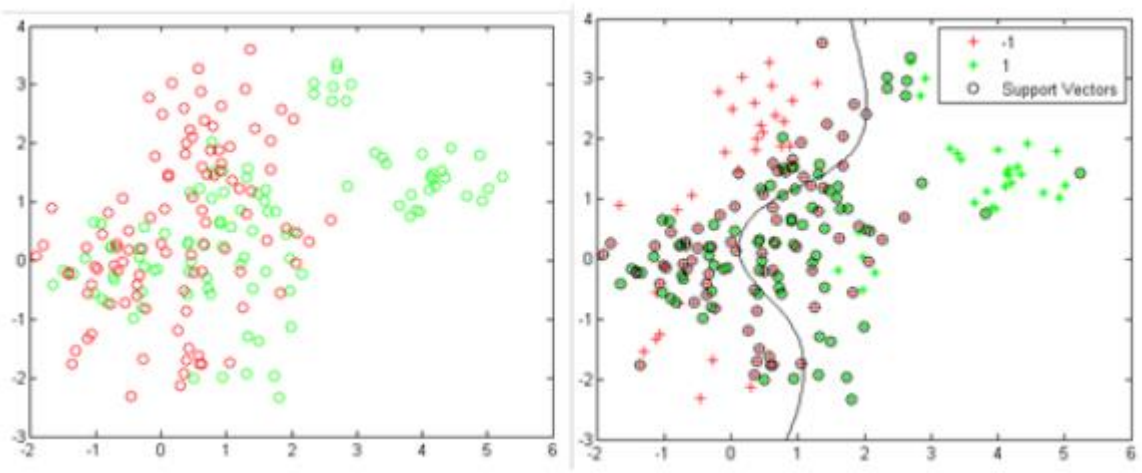
Для класифікації і співвіднесення кожного пікселя до класу дефектів або до класу об'єктів використовується метод опорних векторів (SVM). Для пояснення роботи методу опорних векторів будемо розглядати задачу класифікації для об'єктів двох класів.

У методі опорних векторів виділяють два етапи: етап навчання та етап розпізнавання. На першому етапі з безлічі навчальних прикладів відбираються опорні вектори, на основі яких будується розділяє площину. Етап розпізнавання полягає в тому, що на вхід отриманого класифікатора подається приклад X, про класової приналежності якого нічого не відомо. Класифікатор повинен дати відповідь, до якого класу належить вектор X.

У даній роботі для побудови роздільної гіперплощини використовується радіальна базисна функція:

$$k(x, x') = \exp(-\gamma \|x - x'\|^2) \text{ при } \gamma > 0, \quad (6)$$

де x – поточний вектор, x’ – центральний вектор, y – нормалізуючий параметр. Приклад використання даного ядра наведено на рис. 3.4.



а) б)

Рисунок 3.4 – Ілюстрація побудови роздільної гіперплощини: а) – вхідні значення, б) – приклад роздільної площини

Для навчання SVM класифікатора використовуються 500 зображень дефектів, і 500 зображень без дефектів розміром 20 на 20 пікселів.

Результат роботи запропонованого методу наведено на рис. 3.5.

Рисунок 3.5 – Приклад роботи запропонованого підходу:
а) – вихідне зображення; б) – попередня обробка; в)– результуюча маска подряпин

Аналіз результатів обробки показує, що найбільш явні подряпини на зображенні коректно виявлені, навіть на складних структурних та текстурних особливостях зображень. Попередня обробка шумозаглушення допомагає зменшити вплив шумовий складової, а також усунути невеликі дефекти.

У висновку можна зробити наступні висновки.

У роботі представлений автоматизований алгоритм виділення дефектів на основі модифікованого оператора локальних бінарних околиць. Для класифікації дескрипторів і поділу на класи використаний метод опорних векторів. Приклади, представлені в роботі, демонструють ефективність алгоритму при виявленні подряпин на складно текстурних зображеннях.

3.2 Обробка монохромних зображень

3.2.1 Основна теорія

Чорно-білі зображення можна представити у вигляді сітки з пікселів. У кожного пікселя є значення яскравості, що лежить в діапазоні від 0 до 255, від чорного до білого(рис. 3.6).

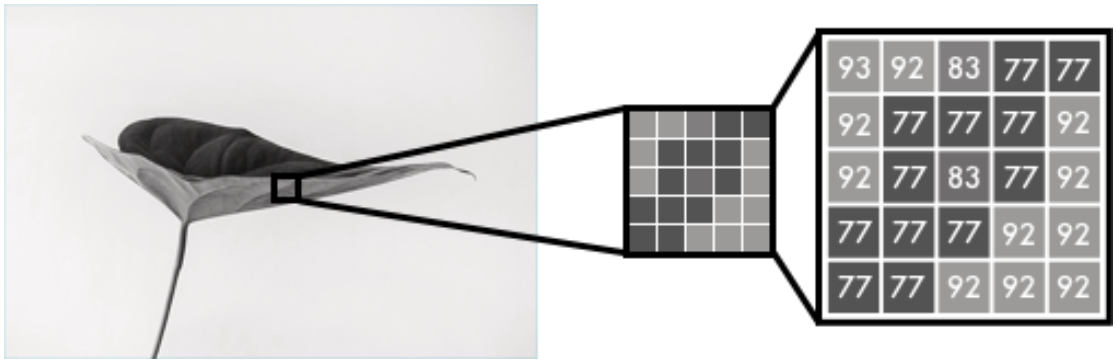


Рисунок 3.6 – Розклад зображення на пікселі зі значеннями яскравості

Кольорові зображення складаються з трьох шарів: червоного, зеленого і синього. Припустимо, потрібно розкласти по трьом каналам картинку з зеленим листком на білому тлі. Він представлений у всіх трьох шарах, тому що шари визначають не тільки колір, але і яскравість. Наприклад, щоб отримати білий колір, нам потрібно отримати рівний розподіл всіх кольорів. Якщо додати однакову кількість червоного і синього, то зелений стане яскравішим. Тобто в кольоровому зображенні за допомогою трьох шарів кодується колір і контрастність(рис 3.7).

Рисунок 3.7 – Кодування різних відтінків одного кольору

Як і в чорно-білому зображенні, пікселі кожного шару кольорового зображення містять значення від 0 до 255. Нуль означає, що у цього пікселя в даному шарі немає кольору. Якщо у всіх трьох каналах стоять нулі, то в результаті на зображенні виходить чорний піксель.

Нейронна мережа встановлює взаємозв'язок між вхідним і вихідним значеннями. У нашому випадку нейромережа повинна знайти сполучні риси між чорно-білими і кольоровими зображеннями. Тобто знайти властивості, за якими можна зіставити значення з чорно-білої сітки зі значеннями трьох кольорових(рис. 3.8).

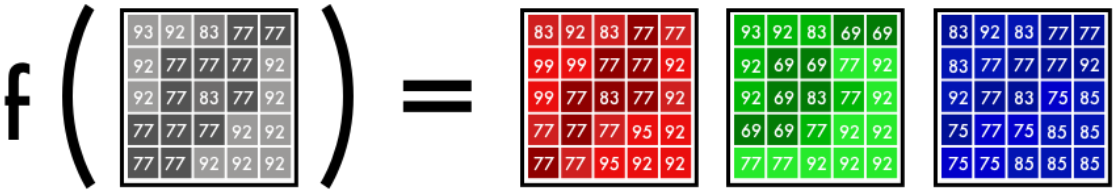


Рисунок 3.8 – Нейронна мережа, вхідні та вихідні дані

3.2.2 Алгоритм роботи з градацією яскравості

Спочатку потрібно скористатися алгоритмом зміни колірних каналів з RGB на Lab. L означає яскравість (lightness), a і b – декартові координати, що визначають положення кольору в діапазоні, відповідно, від зеленого до червоного і від синього до жовтого.

Зображення в просторі Lab містить один шар градацій сірого, а три кольорових шару упаковані в два. Тому ми можемо використовувати в остаточному зображенні вихідний чорно-білий варіант. Залишилося обчислити ще два канали.

В якості вхідних даних слугує шар з градаціями сірого, і на його основі генеруються кольорові шари a і b в колірному просторі Lab. Його ж і потрібно використовувати як L-шар остаточного зображення.

Для отримання двох шарів з одного шару, скористаємося згортковими фільтрами. Їх можна представити як синє і червоне скло в 3D-окулярах. Фільтри визначають, що буде відображатись на картинці. Вони можуть підкреслювати або приховувати якусь частину зображення, щоб око отримувало потрібну інформацію. Нейромережа теж може за допомогою фільтру створити нове зображення або звести кілька фільтрів в одну картинку.

У згорткових нейромережах кожен фільтр автоматично підлаштовується, щоб легше було отримати потрібні вихідні дані. Потрібно накласти сотні фільтрів, а потім звести їх воєдино, щоб отримати шари a і b(рис 3.9).

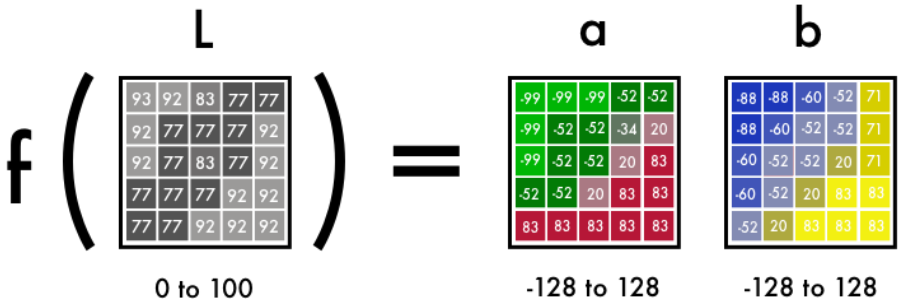


Рисунок 3.9 – Розбиття на a і b слої в кольоровому просторі Lab

Приклад коду для роботи з яскравістю в кольоровому просторі Lab наведено нижче:

```
# Import map images into the lab colorspace
X = rgb2lab(1.0/255*image)[:,:,0]
Y = rgb2lab(1.0/255*image)[:,:,1:]
Y = Y / 128
X = X.reshape(1, 400, 400, 1)
```

```
Y = Y.reshape(1, 400, 400, 2)
model = Sequential()
model.add(InputLayer(input_shape=(None, None, 1)))

# Building the neural network
model = Sequential()
model.add(InputLayer(input_shape=(None, None, 1)))
model.add(Conv2D(8, (3, 3), activation='relu', padding='same',
strides=2))
model.add(Conv2D(8, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(16, (3, 3), activation='relu', padding='same',
strides=2))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
strides=2))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(16, (3, 3), activation='relu', padding='same'))
model.add(UpSampling2D((2, 2)))
model.add(Conv2D(2, (3, 3), activation='tanh', padding='same'))

# Finish model
model.compile(optimizer='rmsprop',loss='mse')

#Train the neural network
model.fit(x=X, y=Y, batch_size=1, epochs=3000)
print(model.evaluate(X, Y, batch_size=1))

# Output colorizations
output = model.predict(X)
output = output * 128
canvas = np.zeros((400, 400, 3))
```

```
canvas[:,:,0] = X[0][:,:,0]
canvas[:,:,1:] = output[0]
```

На вхід подається сітка, що представляє чорно-біле зображення. А на виході – дві сітки зі значеннями кольорів. Між вхідними та вихідними значеннями створюються сполучні фільтри. Це і є згорткова нейронна мережа.

Для навчання мережі використовуються кольорові зображення. Відбуваються перетворення з колірного простору RGB в Lab. Чорно-білий шар подається на вхід, а на виході виходять два розфарбованих шари(рис. 3.10).

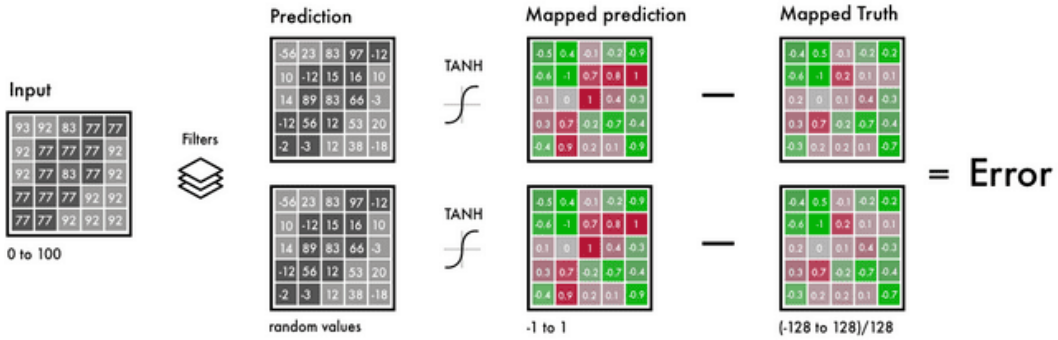


Рисунок 3.10 – Схематичне зображення роботи алгоритму

В одному діапазоні зіставляються *(map)* обчислені значення з реальними, тим самим порівнюючи їх один з одним. Межі діапазону від -1 до 1. Для зіставлення обчислених значень ми використовуємо функцію активації tanh (гіперболічна тангенціальна). Якщо застосувати її до якогось значення, то функція поверне значення в діапазоні від -1 до 1.

Реальні значення кольорів змінюються від -128 до 128. У просторі Lab це діапазон за замовчуванням. Якщо кожне значення розділити на 128, то всі вони виявляться в межах від -1 до 1. Така «нормалізація» дозволяє порівнювати похибку обчислення.

Після обчислення результуючої похибки нейромережа оновлює фільтри, щоб скорегувати результат наступної ітерації. Вся процедура повторюється циклічно, поки похибка не стане мінімальною.

1.0 / 255 означає, що використовується 24-бітовий колірний простір RGB. Тобто для кожного колірного каналу використовується значення в діапазоні від 0 до 255. Це дає 16,7 мільйона кольорів.

Але оскільки людське око може розпізнавати лише від 2 до 10 млн кольорів, то використовувати більш широке колірне простір не має сенсу.

Кольорова палітра Lab використовує інший діапазон. Колірний спектр ab варіюється від -128 до 128. Якщо поділити всі значення вихідного шару на 128, то вони вкладуться в дипазон від -1 до 1, і тоді можна буде зіставити ці значення з тими, що вирахувала нейромережа.

Після того, як за допомогою функції *rgb2lab ()* перетворюється колірний простір, за допомогою *[:,:, 0]* вибирається чорно-білий шар. Це вхідні дані для нейромережі. *[:,:, 1:]* вибирає два кольорових шари, червоно-зелений і синьо-жовтий.

Після навчання нейромережі виконуються останні обчислення, які перетворюються в картинку.

Створюється чорне RGB-полотно, всі три шари заповнюються нулями. Потім копіюється чорно-білий шар з тестового зображення і додається два кольорових шари. Одержаний масив значень пікселів перетворюється в зображення.

3.2.3 Алгоритм узагальнення характеристик

Наша нейромережа шукає характеристики, що зв'язують чорно-білі зображення з їх кольоровими версіями. Алгоритм шукає прості характерні структури: діагональні лінії, тільки чорні пікселі і так далі. У кожному квадрату з 9 пікселів шукається одна і та ж структура і видаляється все, що їй не відповідає. В результаті створюється 64 нових зображення з 64 мініфільтрів(рис 3.11).

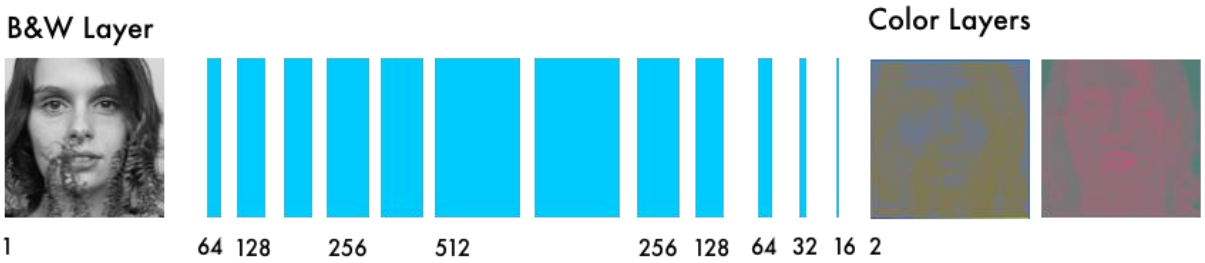


Рисунок 3.11 – Кількість оброблених фільтрами зображень на кожному етапі

Щоб краще проаналізувати зображення, зменшується його розмір удвічі. При застосуванні простіші фільтри до нових квадратів з дев'яти пікселів, то можна виявити більш складні структури. Наприклад, півколо, маленька точка або лінія. Знову раз за разом знаходимо на зображенні одну і ту ж повторювану структуру. На цей раз генеруємо 128 нових оброблених фільтрами зображень (рис. 3.12).

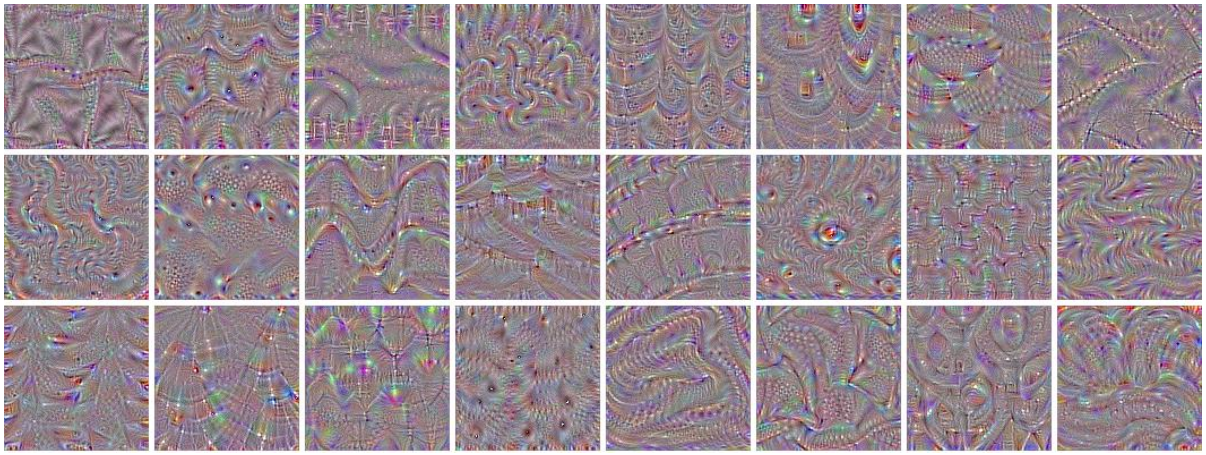


Рисунок 3.12 – Оброблені в декілька єтапів фільтрами зображенння

Описаний процес дуже схожий на алгоритми комп'ютерного зору. Тут ми використовуємо так звану згорткову нейронну мережу, яка об'єднує декілька оброблених зображень, щоб зрозуміти вміст всієї картинки.

Нейромережа діє за принципом проб і помилок. Спочатку вона випадковим чином призначає колір кожного пікселя. Потім по кожному пікселю обчислює помилки і коригує фільтри, щоб в наступній спробі поліпшити результати.

Нейромережа підлаштовує свої фільтри, відштовхуючись від результатів з найбільшими значеннями помилок. У нашому випадку

нейромережа вирішує, чи потрібно розфарбовувати чи ні, і як розташувати на зображенні різні об'єкти. Спочатку вона фарбує всі об'єкти в коричневий. Цей колір найбільше схожий на всі інші кольори, тому з ним при його використанні виходять найменші помилки.

Через одноманітність навчальних даних нейромережа намагається зрозуміти відмінності між тими чи іншими об'єктами.

Алгоритм роботи нейромережі з узагальненням характеритсик:

```
# Set up training and test data
split = int(0.95*len(X))
Xtrain = X[:split]
Xtrain = 1.0/255*Xtrain

#Design the neural network
model = Sequential()
model.add(InputLayer(input_shape=(256, 256, 1)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same',
strides=2))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same',
strides=2))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same',
strides=2))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))

model.add(UpSampling2D((2, 2)))

# Finish model
model.compile(optimizer='rmsprop', loss='mse')

# Image transformer
datagen = ImageDataGenerator(
```

```
shear_range=0.2,
zoom_range=0.2,
rotation_range=20,
horizontal_flip=True)

# Generate training data
batch_size = 50
def image_a_b_gen(batch_size):
for batch in datagen.flow(Xtrain, batch_size=batch_size):
lab_batch = rgb2lab(batch)
X_batch = lab_batch[:,:,:,0]
Y_batch = lab_batch[:,:,:,1:] / 128
yield (X_batch.reshape(X_batch.shape+(1,)), Y_batch)

# Train model
TensorBoard(log_dir='/output')
model.fit_generator(image_a_b_gen(batch_size), steps_per_epoch=10000,
epochs=1)

# Test images
Xtest = rgb2lab(1.0/255*X[split:])[:,:,:,0]
Xtest = Xtest.reshape(Xtest.shape+(1,))
Ytest = rgb2lab(1.0/255*X[split:])[:,:,:,1:]
Ytest = Ytest / 128
print model.evaluate(Xtest, Ytest, batch_size=batch_size)

# Load black and white images
color_me = []
for filename in os.listdir('../Test/'):
color_me.append(img_to_array(load_img('../Test/'+filename)))
color_me = np.array(color_me, dtype=float)
color_me = rgb2lab(1.0/255*color_me)[:,:,:,0]
color_me = color_me.reshape(color_me.shape+(1,))
```

```
# Test model
output = model.predict(color_me)
output = output * 128

# Output colorizations
for i in range(len(output)):
cur = np.zeros((256, 256, 3))
cur[:,:,0] = color_me[i][:,:,0]
cur[:,:,1:] = output[i]
```

Від інших нейромереж, що працюють із зображеннями, ця відрізняється тим, що для неї важливо розташування пікселів. У розфарбовуючих нейромережах розмір зображення або співвідношення сторін залишається незмінним. А у мереж інших типів зображення спотворюється в міру наближення до остаточної версії.

Шар пулінгу з функцією максимуму, застосовуваний в класифікуючих мережах, збільшує щільність інформації, але при цьому спотворює картинку. Він оцінює тільки інформацію, а не макет зображення. А в розфарбовуючих мережах для зменшення ширини і висоти вдвічі використовується крок 2 (*stride of 2*). Щільність інформації теж збільшується, але картинка не спотворюється(рис. 3.13).

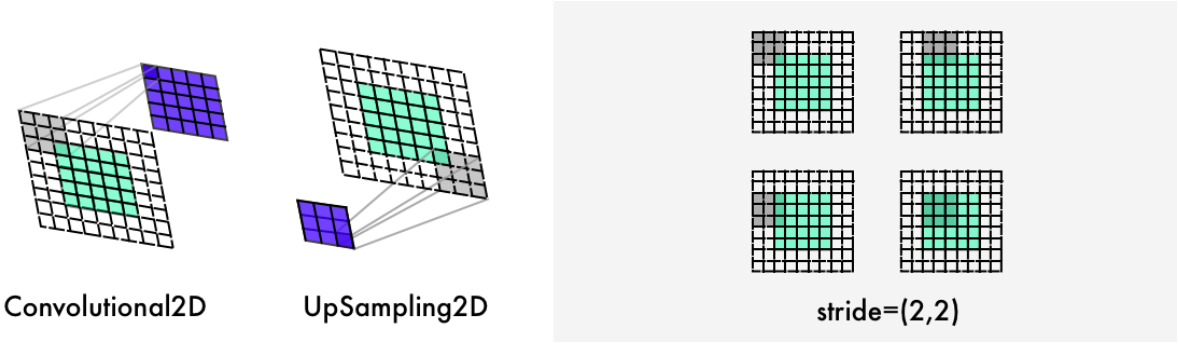


Рисунок 3.13 Приклад роботи нейромережі

Також нейромережа відрізняється від інших шарами підвищення дискретизації (*upsampling*) і збереженням співвідношення сторін зображення. Класифікуючі мережі дбають тільки про підсумкову класифікацію, тому

поступово зменшують розмір і якість картинки в міру її прогону через нейромережу.

Розфарбовуючи нейромережі не змінюють співвідношення сторін зображення. Для цього за допомогою параметра * *padding = 'same'* * додаються білі поля(рис. 3.13). В іншому випадку кожен згортковий шар обрізав би зображення.

Щоб подвоїти розмір картинки, розфарбовуюча нейромережа використовує шар підвищення дискретизації.

Вхідні дані одночасно проходять через кодувальник і через найпотужніший сучасний класифікатор - *Inception ResNet v2*. Це бібліотека з реалізованою нейромережою-класифікатором, що була навчена на 1,2 млн зображень. Витягується шар класифікації і об'єднується з вихідними даними кодувальника(рис. 3.14).

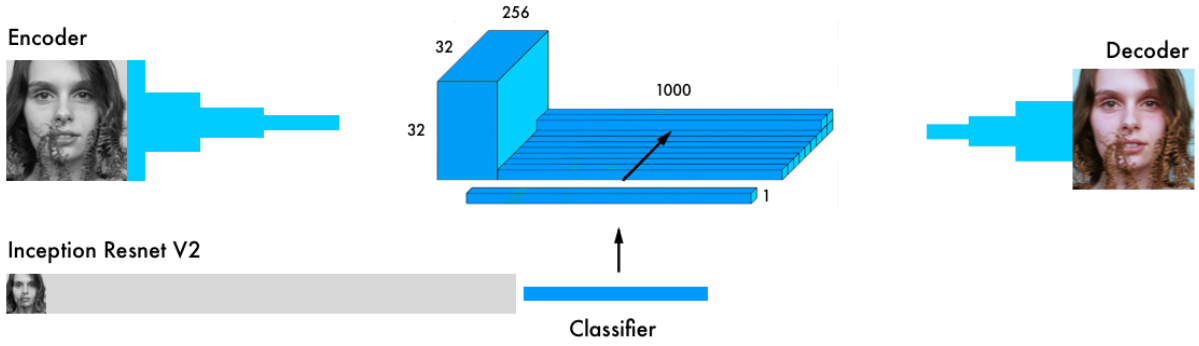


Рисунок 3.14 – Схематичне зображення алгоритму роботи чотирьохкомпонентної нейромережі

Нейромережа зможе зрозуміти, що зображено на картинці, а це означає що зможе і зіставляти уявлення об'єкта зі схемою розфарбовування. Для цього потрібно використати бібліотеку *Inception ResNet v2* і завантажити з неї значення ваг.

```
inception = InceptionResNetV2(weights=None, include_top=True)
inception.load_weights('/data/inception_resnet_v2_weights_tf_dim_order
ing_tf_kernels.h5')
inception.graph = tf.get_default_graph()
```

Створимо серію (*batch*) з підправлених зображень. Переведемо їх в ч / б і проженемо через модель *Inception ResNet*.

```
grayscaled_rgb = gray2rgb(rgb2gray(batch))
embed = create_inception_embedding(grayscaled_rgb)
```

Спочатку потрібно змінити розмір картинок, щоб згодувати їх моделі. Потім за допомогою препроцесора наповнити пікселі і значення кольором до потрібного формату. І нарешті прожгнати зображення через мережу Inception і отримати підсумковий шар моделі.

```
grayscaled_rgb_resized = []
for i in grayscaled_rgb:
i = resize(i, (299, 299, 3), mode='constant')
grayscaled_rgb_resized.append(i)
grayscaled_rgb_resized = np.array(grayscaled_rgb_resized)
grayscaled_rgb_resized = preprocess_input(grayscaled_rgb_resized)
with inception.graph.as_default():
embed = inception.predict(grayscaled_rgb_resized)
return embed
```

encoder_inputis передається в модель *Encoder*, її вихідні дані потім об'єднуються в шарі злиття з *embed_inputin*; вихідні дані злиття подаються на вхід моделі *Decoder*, яка повертає підсумкові дані – *decoder_output*.

```
model = Model(inputs=[encoder_input, embed_input],
outputs=decoder_output)
```

В шарі злиття спочатку шар від 1000 категорій (*1000 category layer*) множиться на 1024 (32 * 32). Так отримується з моделі *Inception* тисячі двадцять чотири ряди підсумкового шару. Сітка 32 х 32 перекладається з двомірного в тривимірне уявлення, від 1000 стовпців категорій (*category pillars*). Потім стовпчики зв'язуються з вихідними даними моделі кодувальника. Застосовуємо згорткову мережу з 254 фільтрами і ядром 1х1 до остаточних результатів шару злиття.

```
fusion_output = RepeatVector(32 * 32)(embed_input)
fusion_output = Reshape(([32, 32, 1000]))(fusion_output)
fusion_output = concatenate([fusion_output, encoder_output], axis=3)
fusion_output = Conv2D(256, (1, 1), activation='relu')(fusion_output)
```

Висновки по розділу

У даному розділі детально описано архітектуру роботи системи колоризації зображення та її підсистем, а саме було реалізовано підсистему для первинної обробки вхідного зображення для усунення дефектів і шумів наявних на ньому та було реалізовано підсистему задачею якої є саме розфарбування пройденого первинну обробку зображення, реалізовано інтерфейс користувач, описані етапи роботи алгоритмічного забезпечення, визначені вимоги до технічного забезпечення, описані програмні засоби, що використовувалися. Зображено процес взаємодії з користувачем та інтерфейс. Підведені підсумки по результатам проведеної роботи.

РОЗДІЛ 4. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ

4.1 Опис ідеї проекту

Ідея проекту полягає у створенні інтелектуальної системи обробки зображень. Призначення системи – обробка вхідних зображень. А саме:

– детектування та усунення наявних дефектів зображення;

– усунення шумів зображення, що можуть виникати при оцифруванні;

– обробка монохромного зображення та його розфарбування;

Таблиця 4.1. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Робочий додаток у якому є можливість обробляти вхідні фотографії та архіви з зображеннями. Додаток також можна використовувати для поліпшення якості зображення.	Усунення дефектів ташуму на зображеннях.	Швидкодіючий алгоритм для покращення якості зображення.
	Реставрація старовинних фото.	Оновлення та оцифрування фотографій що зберіглись від минулого покоління.

На ринку існують аналоги подібних систем, але більшість з них надаються з недостатнім функціоналом та в залежності від вибраного підходу,вони можуть бути як недостатньо якісним, так і занадто повідьними. Ці аналоги в основному англомовні. До того ж розроблена система універсальна та може бути пристосована до різних задач. Тому доцільно проводити аналіз потенційних техніко-економічних переваг ідеї порівняно з пропозиціями конкурентів.

Таблиця 4.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї
1.	Швидкість роботи
2.	Зручність використання
3.	Вимоги до системи
4.	Кросплатформність

Таблиця 4.3. Головні конкуренти

№	Продукція конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
	Мій проект	Colorize Photos	Колор			
1	Швидка	Середня	Повільна	Зв'язок з мережею	Простота архітектури	Потрібно менше знань для виконання заданої задачі
2	Зручно	Зручно	Зручно	Для розфарбування відеопотоку потрібне стороннє розбиття на кадри	Розвиток UI має перспектив у через наявність API	Автоматизована робота
3	Мінімальні	Мінімальні	Високі	Відсутня оптимізація зі старими системами	Простота системи	Актуальність програми для нових систем через використання новітніх бібліотек
4	Наявна	Відсутня	Відсутня	Відсутня оптимізація для роботи	Налаштована система для роботи через головні вікна	–

Ідея проекту є актуальною, можна виділити вагомі переваги для споживачів системи. Перелічені техніко-економічні характеристика, слабкі,

нейтральні та сильні сторони дають підставу вважати, що проект може мати успіх.

4.2 Технологічний аудит ідеї проекту

Для проведення технічного аудиту ідеї проекту, потрібно провести аудит технологій, за допомогою яких можна реалізувати ідею проекту. І для початку потрібно визначити можливість технологічної здійсненності проекту.

Обрана технологія доступна, не потребує доробки, а також безкоштовна та надає усі необхідні можливості для реалізації поставленої задачі. Для розробки з використанням даної технології необхідно мати персональний комп’ютер для можливості встановлення робочого середовища.

Таблиця 4.4. Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Розфарбування монохромних зображень	Використання згорткової нейронної мережі	Розроблені бібліотеки для роботи нейронної мережі	Повністю відкрита для роботи з інформацією
2	Використання існуючих сервісів для обробки зображення	API для роботи з обробки зображень	Розроблені бібліотеки для роботи нейронної мережі	Повністю відкритий програмний код та використання готових бібліотек
Обрана технологія реалізації ідеї проекту:1				

Висновок: технологічна реалізація продукту – можлива, вибрана технологія №1 яка може нам допомогти розробити якісний продукт з використанням комбінації технологій, та перспективи у майбутніх ідей.

4.3 Аналіз ринкових можливостей запуску стартап-проєкту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів. Для цього спочатку проводиться аналіз попиту.

Висновок: враховуючи кількість головних гравців по ринку, зростаючу динаміку ринку, невелику кількість конкурентів та середню норму рентабельності можна зробити висновок, що на даний момент, ринок для входження стартап-продукту є привабливим.

Таблиця 4.5. Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн./ум. од	200
3	Динаміка ринку	Темпи розвитку світової економіки позитивні, але з ознаками зменшення росту.
4	Наявність обмежень для входу	Відсутні. Так як аналоги використовують не автоматизовані алгоритми обробки фотографій
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні.
6	Середня норма рентабельності в галузі або по ринку, %	70

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (табл. 4.6). Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають.

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (табл. 4.7) за моделлю п’яти сил М. Портера, яка вирізняє п’ять основних факторів, що впливають на привабливість вибору ринку з огляду на характер конкуренції:

– конкурент, що вже є у галузі;
– потенційні конкуренти;
– наявність товарів-замінників;
– постачальники, що конкурують за ринкову владу;
– споживачі, які конкурують за ринкову владу.

Таблиця 4.6. Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару
1	Потреба в зменшенні витрат в часі	Старше покоління	Стартап буде переважно буде економити велику кількість часу	Зручність у використанні. Швидка робота системи. Спроможність швидко освоїти як користуватися системою. Можливість редагувати не вірні результати роботи нейронної мережі.
2	Потреба в зменшенні витрат в часі та швидкої публікації фотографій	Звичайні користувачі	Стартап буде переважно буде економити кількість часу та швидкої публікації фотографій в соціальні мережі	

Визначена характеристика дозволяє зробити висновок, що проект знайде свого покупця, а за рахунок подальшого масштабування та адаптації проект можу стати універсальним рішенням.

Таблиця 4.7. Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренти	Наявність конкурентів котрі надають схожі рішення	Зменшення ціни на поставлену послугу; Розробка унікальних характеристик товару; Надання ліцензій на обслуговування
2	Кошти на розробку та підтримку продукту	Закінчення грошей та недостатнє фінансування	Залучення додаткових інвесторів, мотивація роботи на перспективу; Ітеративна розробка
3	Вихід аналогу	Вихід аналогу даного товару може призвести до знецінення та безідейності даного товару	Вихід товару на ринок в коротші строки з не повною, але достатньою, функціональністю для зацікавлення усіх цільових аудиторій; Проведення рекламної компанії

Таблиця 4.8. Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Новий продукт	Вихід на ринок, Зменшення монополії, Надання нових рішень у сфері	Розробка нової функціональності; Вихід нової продукції на ринок; Надання різноманітних типів ліцензій в залежності від потреб користувача \ замовника.
2	Вихід аналогу	Надати продукт з певними характеристиками та можливостями що відсутні у компаній конкурентів	Аналіз ринку та користувачів задля задоволення їх потреб та надання функціональності у найкоротші строки за ціну, котра є дешевшою ніж у продуктів-замінників.
3	Зворотній зв’язок від користувачів	Можливість отримання необхідної інформації для вдосконале	Наявність вхідних даних та реакція на них з боку команди розробників задля задоволення п

Стартап-проект можна впроваджувати на ринок.

Таблиця 4.9. Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1	Тип конкуренції: монополістична	Товар від кожної компанії на ринку, являється недосконалим замінником товару, реалізованого іншими фірмами; На ринку є умови для входу та виходу; Ціна корелює між суперниками;	Розробка продукту з характеристиками, які покривають сфери вживання що не покривають інші товари-замінники; Кореляція цін у відповідності до товарів замінників; Різни типи ліцензій.
2	Рівень конкурентної боротьби: світовий	Всі продукти замінники розроблялись інтернаціональними командами з різних куточків світу, продукти не належать до певної держави, а належать команді розробників	Вихід на ринок збуту продукту з необхідною функціональністю; Налагодження маркетингу на основних Інтернет ресурсах задля охоплення великої кількості потенційних користувачів; Надання бета-версій продукту.
3	Галузева ознака: внутрішньогалузева	Даний тип продукту може використовуватися тільки у сфері розробки IT додатків \ продуктів	Надання зручного, інтуїтивно зрозумілого інтерфейсу; Підтримка всім відомих методів взаємодії з середовищем розробки; Наявність документації та он-лайн підтримки.

Надалі проводиться аналіз пропозиції – визначаються загальні риси конкуренції на ринку (табл. 4.9): визначаються тип можливої майбутньої конкуренції та її інтенсивність, рівень конкурентоспроможності за рівнем конкурентної боротьби, видами товарів і галузевою ознакою.

Проаналізувавши можливості роботи на ринку з огляду на конкурентну ситуацію можна зробити висновок: оскільки кожний з існуючих продуктів не впливає у великій мірі на поточну ситуацію на ринку в цілому, кожний з існуючих продуктів має свою специфічну сферу використання та свої позитивні та негативні сторони щодо рішення певних типів задач, то робота та вихід на даний ринок є можливою і реалізованою задачею. Для виходу на ринок продукт повинен мати функціонал що відсутній у продуктів-аналогів,

повинен задовольняти потреби користувачів, мати необхідний та достатній функціонал з конфігурування, підтримку зі сторони розробників та можливість розробки спеціального функціоналу за відповідною ліцензією.

Таблиця 4.10. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Аналогічні системи на ринку	Нові системи на ринку	Розробники бібліотек, фреймворків та алгоритмів, які можна використовувати у розробках.	Можливість гнучкого впровадження системи.	Аналогічні системи
Висновки	Прямі конкуренти намагаються сконцентруватися на інших напрямках свої продуктів.	На даний момент їх додаток який може дати нам конкуренцію не доробленний і не впроваджений.	Постачальники диктують умови збереження даних, які захищають приватність. Також постачальники не дають змогу зловживати нормами.	Клієнти можуть диктувати умови на ринку через повідомлення на форумах бо в полі відгуків в точках продажу додатку.	Можливість введення стандартизації роботи алгоритмів розпізнавання рухомих об'єктів

Висновок: проект може бути впроваджений на ринку з огляду на конкурентну ситуацію.

Таблиця 4.11. Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Прагматичність	Через запуск стартапу система буде не дуже складно з точки зору архітектури перший час. Через певний період із додаванням функціоналу та оптимізації алгоритмів роботи програмний код буде все складнішим. Такий етап наступить не раніше одного року постійної роботи над проектом.
2	Зручність	Оскільки стартап розробляється на багатьох платформах з різною шириною екранів, то зручність використання системи на різних пристроях буде відігравати не малу роль у спроможності конкурувати з іншими гравцями ринку.

Закінчення таблиці 4.11.

3	Швидкість роботи	Швидкість роботи відіграє велику роль для користувачів, оскільки вони не будуть готові чекати декілька хвилин на виведення результату роботи додатку.
4	Оптимізація	Якщо додаток буде дуже часто видавати помилки при роботі, то користувачі не будуть вважати додаток надійним.
5	Налаштування під користувача	Різні люди мають різні звички, які вони використовують, наприклад, якщо є люди, які люблять працювати за додатком де є темні кольори, а є такі люди, які люблять світлі кольори. Можливість редагувати зовнішній вигляд додатку надає значну перевагу серед конкурентів.
6	Приватність	В останні роки приватність людей та інформація щодо них все частіше зловживається шахраями або великими корпораціями, які потребують погодження з умовами доступу до приватної інформації та її обробки.
7	Технічна підтримка	Якщо технічна підтримка компанії буде працювати своєчасно та швидко, то це допоможе зберегти репутацію компанії на відміну від конкурентів, де їй не приділяють увагу.

Таблиця 4.12. Порівняльний аналіз сильних та слабких сторін системи обробки зображень

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	+1	+2	+3
1	Прагматичність	15	Наш			К1		К2	
2	Зручність	18		К1	Наш		К2		
3	Швидкість роботи	20	К1			К2			Наш
4	Оптимізація	15	К2	Наш	К1				
5	Налаштування під користувача	18			К2	Наш	К1		
6	Приватність	20		К1	К2				Наш
7	Технічна підтримка	17	К2	К1			Наш		

Інтелектуальна система обробки зображень вище завдяки ціновому фактору та новизні запропонованого рішення.

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (Strength, Weak, Opportunities, Troubles) (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін.

Проведений SWOT-аналіз показав, що стартап-проект доцільно реалізовувати.

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів.

Таблиця 4.13. SWOT аналіз стартап-проекту

Сильні сторони (S): – прагматичність системи через її легкість роботи; – простота у використанні; – зменшення затрат у редагуванні; – наявність відкритого вихідного коду; – збереження приватності інформації користувача.	Слабкі сторони (W): – неоптимізованість алгоритму; – швидкість роботи системи;
Можливості (O): – Зворотній зв’язок з клієнтурою компанії для спроможності розвивати проект в інші напрямки.	Загрози (T): – Автономна робота алгоритмів на електронному пристрої; – Складність роботи алгоритму при невиявлених випадках використання додатку.

Таблиця 4.14. Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Безкоштовне надання певного функціоналу у користування споживачам на обмежений термін	Головний ресурс – люди, даний ресурс - наявний	2-3 місяці
2	Реклама	Залучення власних коштів для реклами товару	1-2 місяці
3	Написання статей та опис товару на відомих ресурсах	Головний ресурс – час, даний ресурс - наявний	2-3 тижні
4	Презентація товару на хакатонах й інших ІТ заходах	Ресурс – час та гроші для участі, наявні	1-3 місяці

Є можливість ринкової комерціалізації проекту, наявний попит та рентабельність роботи на ринку.

Є перспективи впровадження з огляду на потенційні групи клієнтів, бар'єри входження, стан конкуренції, конкурентоспроможність проекту. Доцільною є подальша імплементація проекту.

4.4 Розроблення ринкової стратегії проекту

Таблиця 4.15. Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Орендатори	Потребують	Попит є	Присутня	Помірно
2	Продавці	Потребують	Попит є, проте нижчий ніж у орендаторів	Присутня	Помірно

Закінчення таблиці 4.15.

3	Орендарі	Потребують	Попит є	Присутня	Помірно
4	Покупці	Потребують	Попит є, проте нижчий ніж у орендарів	Присутня	Помірно

Які цільові групи обрано: оскільки різниця між цільовими групами зовсім незначна, а також враховуючи той факт, що компанія має бажання почати отримання прибутку якомога швидше, то доцільно враховувати усі цільові групи, тобто використовувати масовий маркетинг, пропонуючи стандартизовану програму.

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку.

Таблиця 4.16. Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Надання функціональності що відсутня у товарів-замінників, підтримка клієнтів.	Проведення реклами, освітлення унікальної функціональності через інтернет ресурси та інші канали, контакт напряму з споживачами; формування лояльності і прихильності споживачів.	Зниження ступеню замінності товару; прихильність клієнтів; відмітні властивості товару; відмітні характеристики товару.	Стратегія диференціації (допускається стратегія спеціалізації).

Таблиця 4.17. Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є товари-замінники, але дані товари замінники не мають деякого необхідного функціоналу	Так, ціль компанії знайти нових споживачів та, частково, забрати існуючих у конкурентів задля задоволення потреб останніх	Компанія частково копіює характеристики товару конкурента, основна ціль компанії розробка нового унікального функціоналу, з підтримкою основного функціоналу конкурентів	Стратегія виклику лідера

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку або проект.

Таблиця 4.18. Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
Доступна ціна, простота і зручність використання, універсальність	Стратегія диференціації	Вирішення важливих поставлених задач швидко, легко та зрозуміло навіть без інструкцій. Легкість і простота у використанні. Доступність через ціну та технічні характеристики.	стандарти якості; метрики ПЗ; ASQAS – automated system of quality assessment software.

Результатом є узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначатиме напрями роботи стартап-компанії на ринку. Отже, робота стартап-компанії на ринку повинна бути спланована орієнтовано

таким чином: за стратегією диференціації виконаний і буде поширюватись товар відмінний за властивостями від своїх аналогів, дотримуючись у конкурентній поведінці стратегії «виклику лідера», тобто випускається один товар для усіх можливих споживачів.

Надалі розроблена трирівнева маркетингова модель товару: уточняються ідея продукту, його фізичні складові, особливості процесу його надання.

4.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач.

Таблиця 4.19. Визначення ключових переваг концепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
Інтелектуальна система обробки зображень.	Отримання з зображень об’єктів їх тривимірну модель та лінійні розміри.	Розрахункові показники, точність та достовірність яких можна оцінювати; простота, кількість вхідних параметрів; самостійність програмної системи.

Таблиця 4.20. Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
Товар за задумом	Створення користувацьких плагінів; інтеграція з популярними CAD системами; локальна обробка зображень; хмарна обробка зображень; створення та редагування тривимірних моделей по зображенню; аналітичний блок оцінки параметрів об’єкта.
Реалізований товар	Підтримка в режимі роботи дизайнера та будівельника; перегляд списку роботів; хмарна обробка зображень; перегляд статусу роботів; створення тривимірних моделей по зображенню.

За рахунок чого потенційний товар буде захищено від копіювання: від копіювання потенційний товар захистити не складає проблеми. Розроблена математична модель підбору пропозицій, на якій базується програмна система,

публікувалась лише у загальних рисах, а без математичної моделі цей ПП лише набір рядків коду.

Визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар, яке передбачає аналіз ціни на товари-аналоги, а також аналіз рівня доходів цільової групи споживачів.

Таблиця 4.21. Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1000 грн	900 грн	10000 грн	500-900 грн

Таблиця 4.22. Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Бажання отримати більше можливостей за менші гроші	Залучення клієнтської бази та продаж	Нульовий рівень: тільки виробник	Вертикальна маркетингова система

Таблиця 4.23. Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Бажання отримати більше за менші гроші	Будь-які, але бажано з великою кількістю візуального контенту	Низька ціна Широкий вибір функціоналу Легкий і простий у використанні продукт	Донести до користувача суть продукту, його якість, та залучити якомога більше зацікавлених клієнтів	Донести до користувача суть продукту, його якість, та залучити якомога більше зацікавлених клієнтів

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 4.21): проводити збут власними силами або залучати сторонніх посередників, вибір та обґрунтування оптимальної глибини каналу збуту, вибір та обґрунтування виду посередників.

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.22).

Як результат було створено ринкову (маркетингову) програму, що включає в себе визначення ключових переваг концепції потенційного товару, опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

Висновки по розділу

В четвертому розділі описано стратегії та підходи з розроблення стартап-проекту, визначено наявність попиту, динаміку та рентабельність роботи ринку, як висновок було вказано що існує можливість ринкової комерціалізації проекту. Розглянувши потенційні групи клієнтів, бар’єри входження, стан конкуренції та конкурентоспроможність проекту було встановлено що проект є перспективним. Розглянуто та вибрано альтернативу впровадження стартап-проекту та доведено доцільність подальшої імплементації проекту.

Отже, ринкова (маркетингова) програма орієнтовано має бути побудована таким чином:

– розробка продукту;

– вибір сегменту ринку та пошук клієнтів;

– стратегія розвитку – стратегія розподіленості, тобто формування конкурентоспроможності досягається шляхом надання споживачу товару, якого той потребує. На основі детального вивчення середовища споживання розробляється одна або декілька особливих характеристик власного товару;

– стратегія конкурентної поведінки – стратегія виклику лідера, тобто на споживчому ринку націлюватись на всіх можливих споживачів, у тому числі клієнтів фірм-конкурентів. Така стратегія будується за принципом «йти слідом» за лідером ринку. За наступні цілі ставиться можливість обійти лідерів цільового сегменту.

Стан та динаміка ринкового середовища на сьогоднішній день і ще багато років є і будуть залишатись сприятливими для впровадження розробленої системи, а також для її необхідності.

Конкурентні переваги створеного продукту очевидні. На вітчизняному ринку аналогів майже не існує, а існуючі – вкрай низької якості. На міжнародному ринку конкуренція наявна та буде рости, якщо не підтримувати та не розвивати свій продукт.

Також, після проведення аналізів можливого цільового сегменту (споживачів), потреб споживачів та можливого попиту, динаміки ринку та рентабельності роботи на ринку, можна однозначно зробити висновок, що створений проект доцільний до комерціалізації.

Перспективи впровадження з огляду на потенційні групи клієнтів, бар’єри входження, стан конкуренції та конкурентоспроможності проекту – прямі, і тільки доводять можливість впровадження, та не марну розробку створеного продукту.

ВИСНОВКИ

Однією з основних проблем при реставрації монохромних зображень є те, що вони з самого початку поступають зашумлені та з певними дефектами. Для вирішення цієї проблеми, було розроблено алгоритм по детектуванню та усуненню дефектів вхідного зображення та його загального знешумлення. Також користувач сам може вирішувати чому віддавати перевагу, швидкості роботи алгоритму з мінімальним покращенням якості зображення, або ж пожертвувати швидкодією заради отримання максимально якісного результату.

Досліджено архітектуру та технології розробки для побудови системи по реставрації чорно-білого зображення. У результаті обрано використовувати: Visual Studio; C#, WPF – для програмної реалізації та інтерфейсу користувача, мову Python та фреймворк TensorFlow – для розробки нейронної мережі. Використання цих сучасних технологій дозволяє досягти високих показників ефективності системи за мінімальну вартість.

Експериментально доведено, що при поєднанні алгоритмів первинної обробки зображення з алгоритмами його колоризації вся система показує відмінні результати, також система вийшла достатньо гнучкою і може підлаштовуватись в залежності від вимог конкретного користувача, за рахунок того, що саме користувач вирішує на що система має ставити акцент – на швидкодію, або на якість.

Результатом магістерської роботи стала система обробки та колоризації монохромних зображень. Система оснащена зручним користувацьким інтерфейсом та задовольняє усі функціональні вимоги системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 2019 Best Photogrammetry Software [Електронний ресурс] // Режим доступу: https://all3dp.com/1/best-photogrammetry-software/
2. S. Ullman (1979). «The interpretation of structure from motion» (PDF). Proceedings of the Royal Society of London. 203 (1153): 405 – 426.
3. openMVG documentation [Електронний ресурс] // Режим доступу: https://openmvg.readthedocs.io/en/latest/
4. Agisoft Metashape [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Metashape
5. RealityCapture [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/RealityCapture
6. PhotoModeler [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/PhotoModeler
7. 3DF Zephyr [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/3DF_Zephyr
8. Pix4D [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Pix4D
9. What is Photogrammetry? [Електронний ресурс] // Режим доступу: http://www.photogrammetry.com/index.htm
10. 3D data acquisition and reconstruction [Електронний ресурс] // Режим доступу:https://en.wikipedia.org/wiki/3D_data_acquisition_and_object_reconstruction
11. Photogrammetry [Електронний ресурс] // Режим доступу: https://en.wikipedia.org/wiki/Photogrammetry
12. ASPRS online Archived May 20, 2015, at the Wayback Machine
13. Sužiedelytė-Visockienė J, Bagdžiūnaitė R, Malys N, Maliene V (2015). "Close-range photogrammetry enables documentation of environment-induced deformation of architectural heritage". Environmental

Engineering and Management Journal. 14 (6): 1371–1381. doi:10.30638/eemj.2015.149.

14. "Ina Jarve, Natalja Liba. The Effect of Various Principles of External Orientation on the Overall Triangulation Accuracy. TECHNOLOGIJOS MOKSLAI. Estonia. #86, 2010, pp. 59-64" (PDF).
15. Розроблення стартап-проекту [Електронний ресурс] : Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с.
16. Zhou, X., Zhu, M., Leonardos, S., & Daniilidis, K. (2016). Sparse representation for 3D shape estimation: A convex relaxation approach. IEEE transactions on pattern analysis and machine intelligence, 39(8), 1648-1661.
17. Zhang, X.; Jiang, Z.; Zhang, H.; Wei, Q. Vision-based pose estimation for textureless space objects by contour points matching. IEEE Trans. Aerosp. Electron. Syst. 2018, 54, 2342–2355.
18. Cao, Z.; Sheikh, Y.; Banerjee, N.K. Real-time scalable 6DOF pose estimation for textureless objects. In Proceedings of the International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 2441–2448.
19. Xiang, Y.; Mottaghi, R.; Savarese, S. Beyond PASCAL: A benchmark for 3D object detection in the wild. In Proceedings of the Workshop on Applications of Computer Vision, Steamboat Springs, CO, USA, 24–26 March 2014; pp. 75–82.
20. Wu, J.; Xue, T.; Lim, J.J.; Tian, Y.; Tenenbaum, J.B.; Torralba, A.; Freeman, W.T. 3D Interpreter networks for viewer-centered wireframe modeling. Int. J. Comput. Vis. 2018, 126, 1009–1026.
21. Yoshifumi Kitamura and Fumio Kishino, “A Parallel Algorithm for Octree Generation from Polyhedral Shape Representation”, Pattern Recognition, 1996. Proceedings of the 13th International Conference on, pp. 303-309, Vol. 3, 1996.

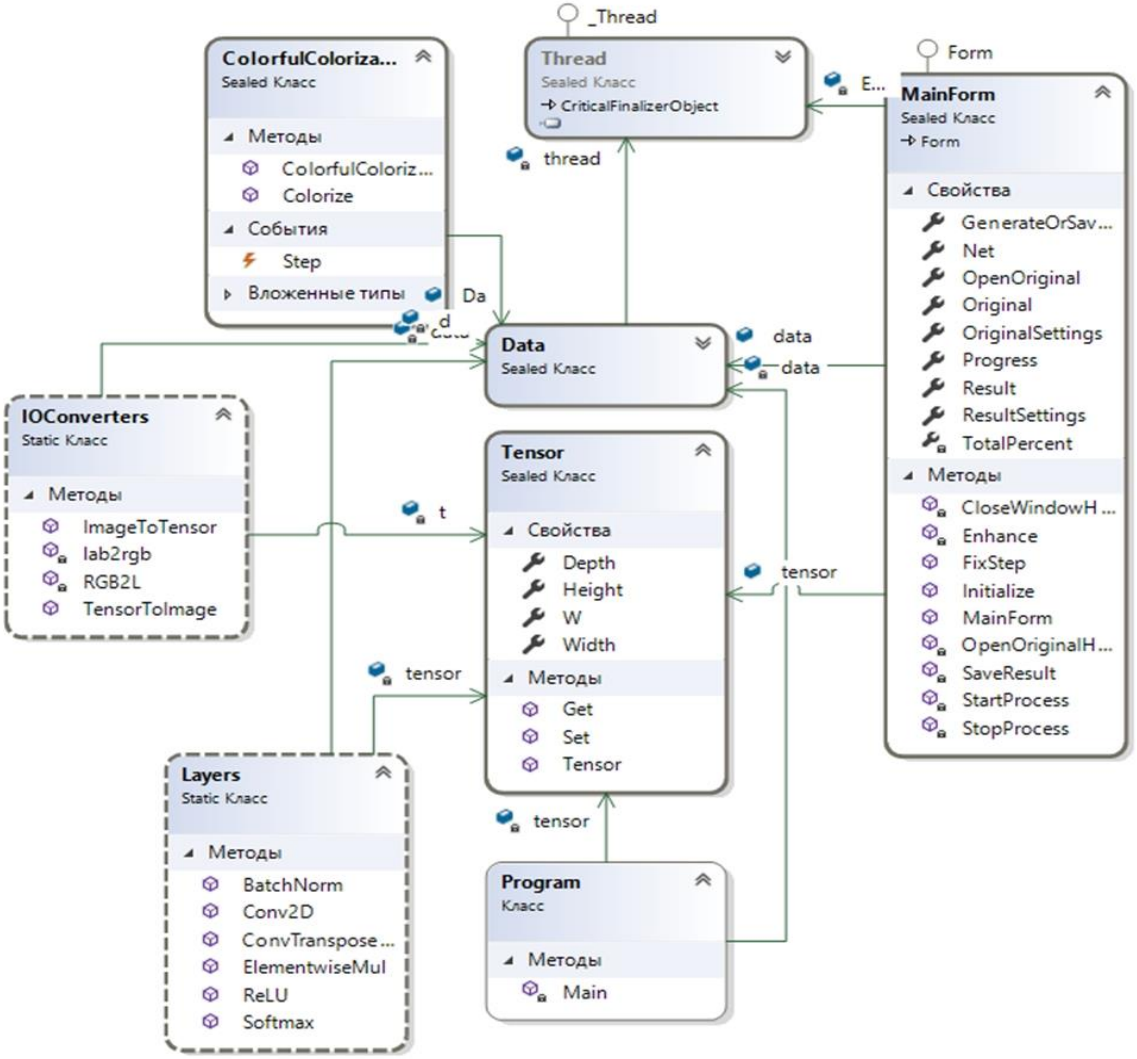
22. Zhang Cha, Chen Tsuhan, “Efficient feature extraction for 2D/3D objects in mesh representation”, In: Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205). IEEE, 2001. p. 935-938.
23. Zhang, X.; Jiang, Z.; Zhang, H.; Wei, Q. Vision-based pose estimation for textureless space objects by contour points matching. IEEE Trans. Aerosp. Electron. Syst. 2018, 54, 2342–2355.
24. Cao, Z.; Sheikh, Y.; Banerjee, N.K. Real-time scalable 6DOF pose estimation for textureless objects. In Proceedings of the International Conference on Robotics and Automation, Stockholm, Sweden, 16–21 May 2016; pp. 2441–2448.
25. Li, C.; Zeeshan Zia, M.; Tran, Q.H.; Yu, X.; Hager, G.D.; Chandraker, M. Deep supervision with shape concepts for occlusion-aware 3D object parsing. In Proceedings of the Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 388–397.
26. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. 2010, 32, 1627–1645.
27. Newell, A.; Yang, K.; Deng, J. Stacked hourglass networks for human pose estimation. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 483–499.

ДОДАТКИ

ДОДАТОК А

Діаграма класів

Діаграма класів



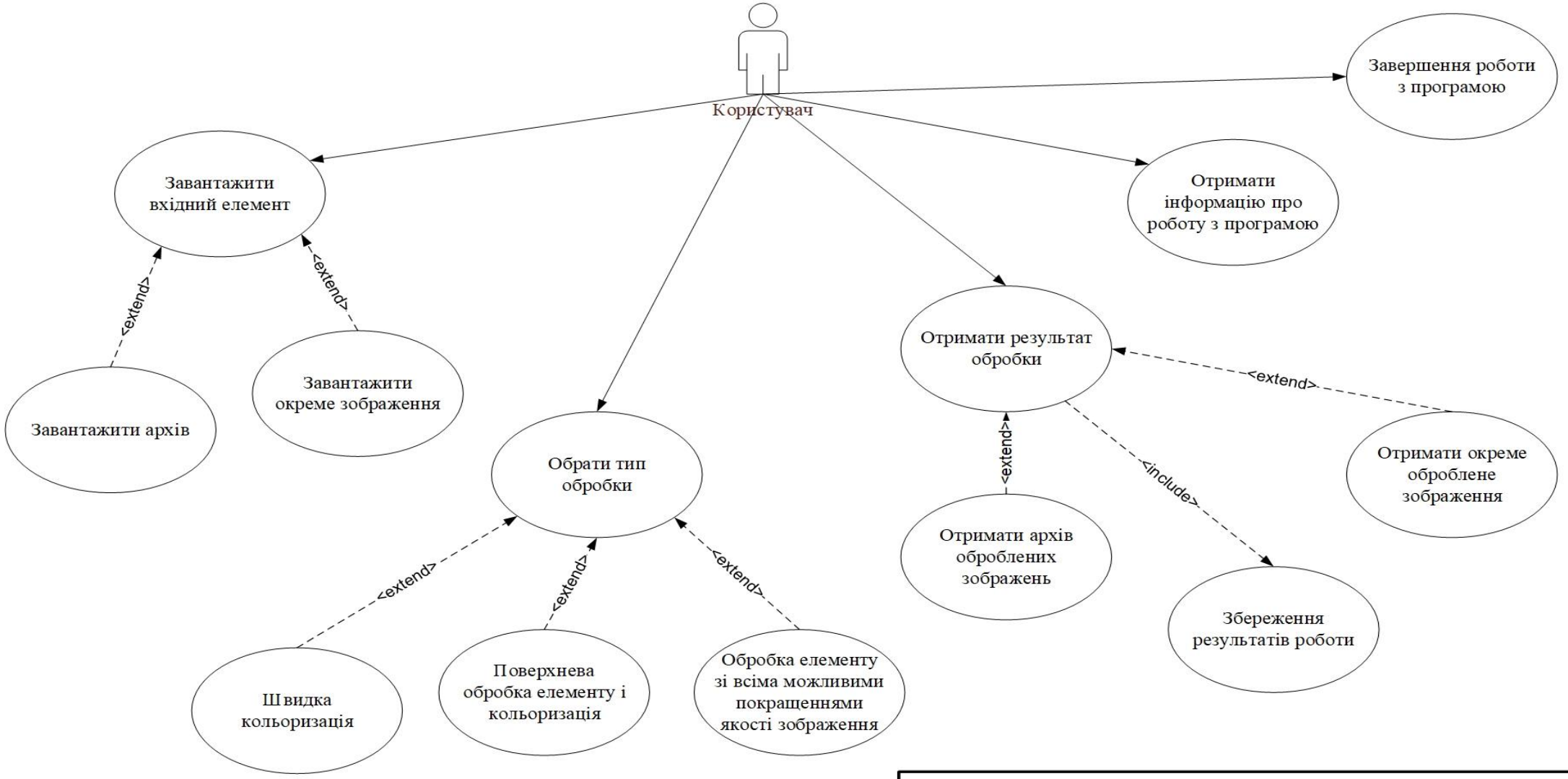
Демонстраційний плакат № 1
до дипломної роботи на тему
„Інтелектуальна система перетворення чорно-білого зображення”

Розробив: _____
Прийняв: _____

ДОДАТОК Б

Діаграма використання системи

Діаграма використання системи



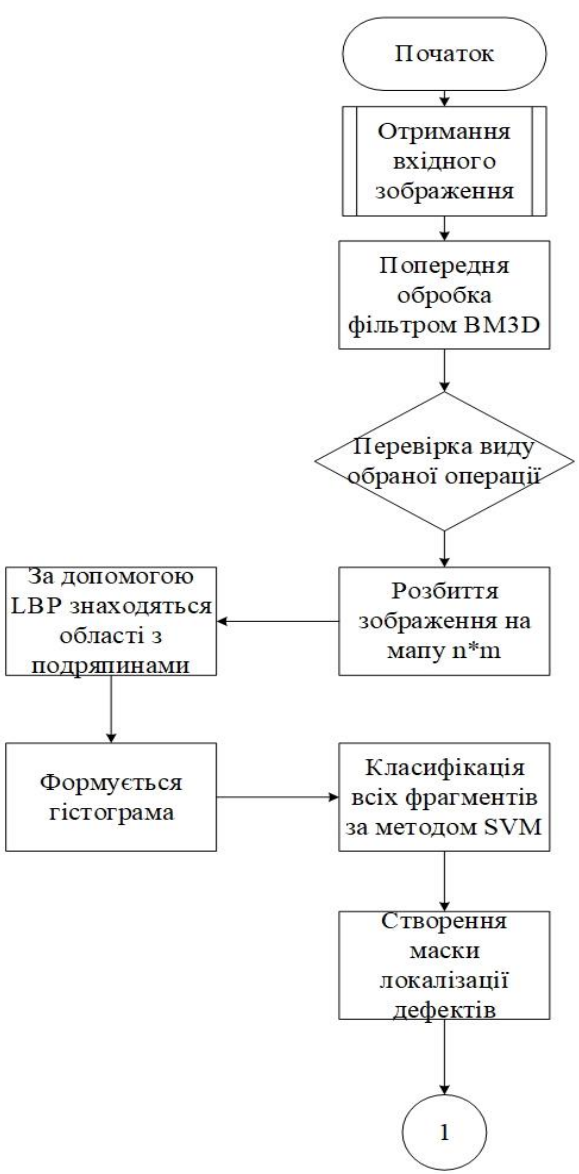
Демонстраційний плакат № 2
до дипломної роботи на тему
„Інтелектуальна система перетворення чорно-білого зображення”

Розробив: _____
Прийняв: _____

ДОДАТОК В

Схема алгоритму усунення дефектів

Схема алгоритму усунення дефектів



Демонстраційний плакат № 3
до дипломної роботи на тему
„Інтелектуальна система перетворення чорно-білого зображення”

Розробив: _____
Прийняв: _____

ДОДАТОК Г

Результат перевірки на співпадіння

ДОДАТОК Д

Лістинг коду програми

```
public static Tensor ConvTranspose2D(Tensor input, Tensor[] Filters, Tensor Biases,
int stride = 2)
        {
            var OutputWidth = input.Width * stride;
            var OutputHeight = input.Height * stride;
            var OutputDepth = Filters[0].Depth;
            var tempOutput = new Tensor(OutputWidth, OutputHeight, OutputDepth);
            System.Threading.Tasks.Parallel.For(0, input.Depth, (int d) =>
            {
                var f = Filters[d];
                var x = -f.Width / 2;
                var y = -f.Height / 2;
                for(var ay = 0; ay < input.Height; y += stride, ay++)
                {
                    x = -f.Width / 2;
                    for(var ax = 0; ax < input.Width; x += stride, ax++)
                    {
                        var chain_grad = input.Get(ax, ay, d);
                        for(int fy = 0; fy < f.Height; fy++)
                        {
                            var oy = y + fy;
                            for(int fx = 0; fx < f.Width; fx++)
                            {
                                var ox = x + fx;
                                if((oy >= 0) && (oy < OutputHeight) && (ox >= 0) &&
(ox < OutputWidth))
                                {
                                    for(int fd = 0; fd < f.Depth; fd++)
                                    {
                                        var ix1 = ((OutputWidth * oy) + ox) *
OutputDepth + fd;
                                        var ix2 = ((f.Width * fy) + fx) * f.Depth +
fd;
                                        tempOutput.W[ix1] += f.W[ix2] * chain_grad;
                                    }
                                }
                            }
                        }
                    }
                }
            });
            Parallel.For(0, tempOutput.Depth, (int d) =>
            {
                for(int y = 0; y < tempOutput.Height; y++)
                {
                    for(int x = 0; x < tempOutput.Width; x++)
                    {
                        tempOutput.Set(x, y, d, tempOutput.Get(x, y, d) +
Biases.W[d]);
                    }
                }
            });
            return tempOutput;
        }

public static Tensor Conv2D(Tensor input, Tensor[] Filters, Tensor Biases, int stride =
1, int dilation = 1)
        {
            var OutputDepth = Filters.Length;
            var Result = new Tensor(input.Width / stride, input.Height / stride,
OutputDepth);
            Parallel.For(0, OutputDepth, (int d) =>
```

```
            {
                var f = Filters[d];
                for (int ay = 0; ay < Result.Height; ay++)
                {
                    var y = ay * stride - (f.Height * dilation + dilation - 1) / 2;
                    for (int ax = 0; ax < Result.Width; ax++)
                    {
                        var x = ax * stride - (f.Width * dilation + dilation - 1) / 2;
                        var a = 0.0f;
                        for (int fy = 0; fy < f.Height; fy++)
                        {
                            var oy = y + fy * dilation + dilation - 1;
                            for (int fx = 0; fx < f.Width; fx++)
                            {
                                var ox = x + fx * dilation + dilation - 1;
                                if ((oy >= 0) && (oy < input.Height) && (ox >= 0) && (ox
< input.Width))
                                {
                                    var fi = ((f.Width * fy) + fx) * f.Depth;
                                    var ti = ((input.Width * oy) + ox) * input.Depth;
                                    for (var fd = 0; fd < f.Depth; fd++)
                                    {
                                        a += f.W[fi + fd] * input.W[ti + fd];
                                    }
                                }
                            }
                        }
                        Result.Set(ax, ay, d, a + Biases.W[d]);
                    }
                }
            });
            return Result;


public static partial class Layers
    {

        public static Tensor BatchNorm(Tensor input, Tensor mean, Tensor variance, float
scale)
        {
            var Normalized = new Tensor(input.Width, input.Height, input.Depth);
            Parallel.For(0, input.Depth, (int d) =>
            {
                for(int y = 0; y < input.Height; y++)
                {
                    for(int x = 0; x < input.Width; x++)
                    {
                        Normalized.Set(x, y, d, (float)(((input.Get(x, y, d) - mean.W[d]
/ scale) / Math.Sqrt(1e-5f + variance.W[d] / scale))));
                    }
                }
            });
            return Normalized;
        }

    }

private static Tensor[] LoadWeights(BinaryReader br, int w, int h, int d, int n)
        {
            var Result = new Tensor[n];
            for(int i = 0; i < n; i++)
            {
                Result[i] = new Tensor(w, h, d);
                var T = Result[i];
```

```
                for(int z = 0; z < d; z++)
                {
                    for(int y = 0; y < h; y++)
                    {
                        for(int x = 0; x < w; x++)
                        {
                            T.Set(x, y, z, br.ReadSingle());
                        }
                    }
                }
            }
            return Result;
        }


public static Tensor ElementwiseMul(Tensor input, float Scale)
        {
            var Height = input.Height;
            var Width = input.Width;
            var Result = new Tensor(input.Width, input.Height, input.Depth);
            Parallel.For(0, input.Depth, (int d) =>
            {
                for(int y = 0; y < Height; y++)
                {
                    for(int x = 0; x < Width; x++)
                    {
                        var v = input.Get(x, y, d);
                        Result.Set(x, y, d, v * Scale);
                    }
                }
            });
            return Result;
        }

public static Tensor ReLU(Tensor input)
        {
            var Height = input.Height;
            var Width = input.Width;
            var Result = new Tensor(input.Width, input.Height, input.Depth);
            Parallel.For(0, input.Depth, (int d) =>
            {
                for(int y = 0; y < Height; y++)
                {
                    for(int x = 0; x < Width; x++)
                    {
                        var v = input.Get(x, y, d);
                        Result.Set(x, y, d, (v > 0.0f) ? (v) : (0.0f));
                    }
                }
            });
            return Result;
        }
X = []
for filename in os.listdir('/data/images/Train/'):
X.append(img_to_array(load_img('/data/images/Train/'+filename)))
X = np.array(X, dtype=float)
Xtrain = 1.0/255*X

#Load weights
inception = InceptionResNetV2(weights=None, include_top=True)
inception.load_weights('/data/inception_resnet_v2_weights_tf_dim_ordering_tf_kern
els.h5')
inception.graph = tf.get_default_graph()
```

```
embed_input = Input(shape=(1000,))

#Encoder
encoder_input = Input(shape=(256, 256, 1,))
encoder_output = Conv2D(64, (3,3), activation='relu', padding='same',
strides=2)(encoder_input)
encoder_output = Conv2D(128, (3,3), activation='relu',
padding='same')(encoder_output)
encoder_output = Conv2D(128, (3,3), activation='relu', padding='same',
strides=2)(encoder_output)
encoder_output = Conv2D(256, (3,3), activation='relu',
padding='same')(encoder_output)
encoder_output = Conv2D(256, (3,3), activation='relu', padding='same',
strides=2)(encoder_output)
encoder_output = Conv2D(512, (3,3), activation='relu',
padding='same')(encoder_output)
encoder_output = Conv2D(512, (3,3), activation='relu',
padding='same')(encoder_output)
encoder_output = Conv2D(256, (3,3), activation='relu',
padding='same')(encoder_output)

#Fusion
fusion_output = RepeatVector(32 * 32)(embed_input)
fusion_output = Reshape(([32, 32, 1000]))(fusion_output)
fusion_output = concatenate([encoder_output, fusion_output], axis=3)
fusion_output = Conv2D(256, (1, 1), activation='relu',
padding='same')(fusion_output)

#Decoder
decoder_output = Conv2D(128, (3,3), activation='relu',
padding='same')(fusion_output)
decoder_output = UpSampling2D((2, 2))(decoder_output)
decoder_output = Conv2D(64, (3,3), activation='relu',
padding='same')(decoder_output)
decoder_output = UpSampling2D((2, 2))(decoder_output)
decoder_output = Conv2D(32, (3,3), activation='relu',
padding='same')(decoder_output)
decoder_output = Conv2D(16, (3,3), activation='relu',
padding='same')(decoder_output)
decoder_output = Conv2D(2, (3, 3), activation='tanh',
padding='same')(decoder_output)
decoder_output = UpSampling2D((2, 2))(decoder_output)
model = Model(inputs=[encoder_input, embed_input], outputs=decoder_output)

#Create embedding
def create_inception_embedding(grayscaled_rgb):
grayscaled_rgb_resized = []
for i in grayscaled_rgb:
i = resize(i, (299, 299, 3), mode='constant')
grayscaled_rgb_resized.append(i)
grayscaled_rgb_resized = np.array(grayscaled_rgb_resized)
grayscaled_rgb_resized = preprocess_input(grayscaled_rgb_resized)
with inception.graph.as_default():
embed = inception.predict(grayscaled_rgb_resized)
return embed

# Image transformer
datagen = ImageDataGenerator(
```

```
shear_range=0.4,
zoom_range=0.4,
rotation_range=40,
horizontal_flip=True)

#Generate training data
batch_size = 20
def image_a_b_gen(batch_size):
for batch in datagen.flow(Xtrain, batch_size=batch_size):
grayscaled_rgb = gray2rgb(rgb2gray(batch))
embed = create_inception_embedding(grayscaled_rgb)
lab_batch = rgb2lab(batch)
X_batch = lab_batch[:,:,:,0]
X_batch = X_batch.reshape(X_batch.shape+(1,))
Y_batch = lab_batch[:,:,:,1:] / 128
yield ([X_batch, create_inception_embedding(grayscaled_rgb)], Y_batch)

#Train model
tensorboard = TensorBoard(log_dir="/output")
model.compile(optimizer='adam', loss='mse')
model.fit_generator(image_a_b_gen(batch_size), callbacks=[tensorboard],
epochs=1000, steps_per_epoch=20)

#Make a prediction on the unseen images
color_me = []
for filename in os.listdir('../Test/'):
color_me.append(img_to_array(load_img('../Test/'+filename)))
color_me = np.array(color_me, dtype=float)
color_me = 1.0/255*color_me
color_me = gray2rgb(rgb2gray(color_me))
color_me_embed = create_inception_embedding(color_me)
color_me = rgb2lab(color_me)[:,:,:,0]
color_me = color_me.reshape(color_me.shape+(1,))

# Test model
output = model.predict([color_me, color_me_embed])
output = output * 128

# Output colorizations
for i in range(len(output)):
cur = np.zeros((256, 256, 3))
cur[:,:,0] = color_me[i][:,:,0]
cur[:,:,1:] = output[i]
imsave("result/img_"+str(i)+".png", rgb(cur))
```

```
public static Tensor Softmax(Tensor input)
        {
            var Height = input.Height;
            var Width = input.Width;
            var Result = new Tensor(input.Width, input.Height, input.Depth);
            Parallel.For(0, input.Height, (int y) =>
            {
                for(int x = 0; x < Width; x++)
                {
                    float amax = float.MinValue;
                    for(int d = 0; d < input.Depth; d++)
                    {
                        var v = input.Get(x, y, d);
```

```
                        if(amax < v)
                        {
                            amax = v;
                        }
                    }
                    double sum = 0.0;
                    for(int d = 0; d < input.Depth; d++)
                    {
                        var v = input.Get(x, y, d);
                        sum += System.Math.Exp((double)v - amax);
                    }
                    for(int d = 0; d < input.Depth; d++)
                    {
                        var v = input.Get(x, y, d);
                        Result.Set(x, y, d, (float)(System.Math.Exp((double)v - amax) /
sum));
                    }
                }
            });
            return Result;
        }

public sealed class ColorfulColorization
    {

        public delegate void StepDone(float percent);

        public event StepDone Step;

        public Data Data;

        public ColorfulColorization(Stream s)
        {
            this.Data = new Data(s);
        }

        public Tensor Colorize(Tensor Input)
        {
            var Temp = Input;
            //**********
            //* Conv1
            //**********
            Temp = Layers.Conv2D(Temp, this.Data.BW_Conv1_1_Weights,
this.Data.BW_Conv1_1_Biases);
            if(this.Step != null)
            {
                this.Step(100f / 55f);
            }
            Temp = Layers.ReLU(Temp);
            if(this.Step != null)
            {
                this.Step(100f / 55f);
            }
            Temp = Layers.Conv2D(Temp, this.Data.Conv1_2_Weights,
this.Data.Conv1_2_Biases, 2);
            if(this.Step != null)
            {
                this.Step(100f / 55f);
            }
            Temp = Layers.ReLU(Temp);
            if(this.Step != null)
            {
                this.Step(100f / 55f);
            }
```

```
            Temp = Layers.BatchNorm(Temp, this.Data.Conv1_2_Mean,
this.Data.Conv1_2_Variance, this.Data.Conv1_2_Scale);
            if(this.Step != null)
            {
                this.Step(100f / 55f);
            }


public static unsafe Tensor ImageToTensor(Bitmap Image)
        {
            Image = new Bitmap(Image, 224, 224);
            var Result = new Tensor(Image.Width, Image.Height, 1);
            var BD = Image.LockBits(new Rectangle(0, 0, Image.Width, Image.Height),
ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);
            var w = Image.Width;
            for(int y = 0; y < Image.Height; y++)
            {
                var Addr = (byte*)(BD.Scan0.ToInt32() + BD.Stride * y);
                for(int x = 0; x < w; x++)
                {
                    var B = *Addr;
                    Addr += 1;
                    var G = *Addr;
                    Addr += 1;
                    var R = *Addr;
                    Addr += 1;
                    Result.Set(x, y, 0, RGB2L(R, G, B) - 50f);
                }
            }
            Image.UnlockBits(BD);
            return Result;
        }


public static unsafe Bitmap TensorToImage(Bitmap l, Tensor ab)
        {
            var tmp = new Bitmap(l, ab.Width, ab.Height);
            var BD = tmp.LockBits(new Rectangle(0, 0, tmp.Width, tmp.Height),
ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb);
            for(int _y = 0; _y < tmp.Height; _y++)
            {
                byte* Addr = (byte*)(BD.Scan0.ToInt32() + BD.Stride * _y);
                for(int _x = 0; _x < tmp.Width; _x++)
                {
                    float r = *(Addr + 2);
                    float g = *(Addr + 1);
                    float b = *(Addr);
                    lab2rgb(RGB2L((byte)r, (byte)g, (byte)b), ab.Get(_x, _y, 0),
ab.Get(_x, _y, 1), ref r, ref g, ref b);
                    *(Addr) = (byte)(System.Math.Min(255f, System.Math.Max(0f, b)));
                    Addr += 1;
                    *(Addr) = (byte)(System.Math.Min(255f, System.Math.Max(0f, g)));
                    Addr += 1;
                    *(Addr) = (byte)(System.Math.Min(255f, System.Math.Max(0f, r)));
                    Addr += 1;
                }
            }
            tmp.UnlockBits(BD);
            tmp = new Bitmap(tmp, l.Width, l.Height);
            var BD_ab = tmp.LockBits(new Rectangle(0, 0, tmp.Width, tmp.Height),
ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb);
            var BD_l = l.LockBits(new Rectangle(0, 0, l.Width, l.Height),
ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);
            for(int y = 0; y < l.Height; y++)
```

```
            {
                byte* Addr_ab = (byte*)(BD_ab.Scan0.ToInt32() + BD_ab.Stride * y);
                byte* Addr_l = (byte*)(BD_l.Scan0.ToInt32() + BD_l.Stride * y);
                for(int x = 0; x < l.Width; x++)
                {
                    var Y = *(Addr_l + 2) * 0.299f + *(Addr_l + 1) * 0.587f + *Addr_l *
0.114f;
                    var U = *(Addr_ab + 2) * -0.169f + *(Addr_ab + 1) * -0.332f +
*Addr_ab * 0.500f + 128.0f;
                    var V = *(Addr_ab + 2) * 0.500f + *(Addr_ab + 1) * -0.419f + *Addr_ab
* -0.0813f + 128.0f;
                    var R = Y + (1.4075f * (V - 128f));
                    var G = Y - (0.3455f * (U - 128f) - (0.7169f * (V - 128f)));
                    var B = Y + (1.7790f * (U - 128f));
                    R = System.Math.Min(System.Math.Max(R, 0f), 255f);
                    G = System.Math.Min(System.Math.Max(G, 0f), 255f);
                    B = System.Math.Min(System.Math.Max(B, 0f), 255f);
            }
            tmp.UnlockBits(BD_ab);
            l.UnlockBits(BD_l);
            return tmp;
        }
```